



# **Logic for Computer Science and Artificial Intelligence**

**Ricardo Caferra**

**ISTE**

 **WILEY**



## Logic for Computer Science and Artificial Intelligence



# **Logic for Computer Science and Artificial Intelligence**

Ricardo Caferra

**ISTE**

 **WILEY**

First published 2011 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Ltd  
27-37 St George's Road  
London SW19 4EU  
UK

John Wiley & Sons, Inc.  
111 River Street  
Hoboken, NJ 07030  
USA

[www.iste.co.uk](http://www.iste.co.uk)

[www.wiley.com](http://www.wiley.com)

© ISTE Ltd 2011

The rights of Ricardo Caferra to be identified as the author of this work have been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

---

Library of Congress Cataloging-in-Publication Data

Caferra, Ricardo, 1945-  
Logic for computer science and artificial intelligence / Ricardo Caferra.  
p. cm.  
Includes bibliographical references and index.  
ISBN 978-1-84821-301-2  
1. Computer logic. 2. Artificial intelligence. I. Title.  
QA76.9.L63C34 2011  
006.3--dc23

2011014705

---

British Library Cataloguing-in-Publication Data  
A CIP record for this book is available from the British Library  
ISBN 978-1-84821-301-2

---

Printed and bound in Great Britain by CPI Antony Rowe, Chippenham and Eastbourne.



## Table of Contents

<b>Preface</b> . . . . .	xi
<b>Chapter 1. Introduction</b> . . . . .	1
1.1. Logic, foundations of computer science, and applications of logic to computer science . . . . .	1
1.2. On the utility of logic for computer engineers . . . . .	3
<b>Chapter 2. A Few Thoughts Before the Formalization</b> . . . . .	7
2.1. What is logic? . . . . .	7
2.1.1. Logic and paradoxes . . . . .	8
2.1.2. Paradoxes and set theory . . . . .	9
2.1.2.1. The answer . . . . .	10
2.1.3. Paradoxes in arithmetic and set theory . . . . .	13
2.1.3.1. The halting problem . . . . .	13
2.1.4. On formalisms and well-known notions . . . . .	15
2.1.4.1. Some “well-known” notions that could turn out to be difficult to analyze . . . . .	19
2.1.5. Back to the definition of logic . . . . .	23
2.1.5.1. Some definitions of logic for all . . . . .	24
2.1.5.2. A few more technical definitions . . . . .	24
2.1.5.3. Theory and meta-theory (language and meta-language) . . . . .	30
2.1.6. A few thoughts about logic and computer science . . . . .	30
2.2. Some historic landmarks . . . . .	32
<b>Chapter 3. Propositional Logic</b> . . . . .	39
3.1. Syntax and semantics . . . . .	40
3.1.1. Language and meta-language . . . . .	43

3.1.2. Transformation rules for cnf and dnf . . . . .	49
3.2. The method of semantic tableaux . . . . .	54
3.2.1. A slightly different formalism: signed tableaux . . . . .	58
3.3. Formal systems . . . . .	64
3.3.1. A capital notion: the notion of proof . . . . .	64
3.3.2. What do we learn from the way we do mathematics? . . . . .	72
3.4. A formal system for PL (PC) . . . . .	78
3.4.1. Some properties of formal systems . . . . .	84
3.4.2. Another formal system for PL (PC) . . . . .	86
3.4.3. Another formal system . . . . .	86
3.5. The method of Davis and Putnam . . . . .	92
3.5.1. The Davis–Putnam method and the SAT problem . . . . .	95
3.6. Semantic trees in PL . . . . .	96
3.7. The resolution method in PL . . . . .	101
3.8. Problems, strategies, and statements . . . . .	109
3.8.1. Strategies . . . . .	110
3.9. Horn clauses . . . . .	113
3.10. Algebraic point of view of propositional logic . . . . .	114
<b>Chapter 4. First-order Terms . . . . .</b>	<b>121</b>
4.1. Matching and unification . . . . .	121
4.1.1. A motivation for searching for a matching algorithm . . . . .	121
4.1.2. A classification of trees . . . . .	123
4.2. First-order terms, substitutions, unification . . . . .	125
<b>Chapter 5. First-Order Logic (FOL) or Predicate Logic (PL1, PC1) . . . . .</b>	<b>131</b>
5.1. Syntax . . . . .	133
5.2. Semantics . . . . .	137
5.2.1. The notions of truth and satisfaction . . . . .	139
5.2.2. A variant: multi-sorted structures . . . . .	150
5.2.2.1. Expressive power, sort reduction . . . . .	150
5.2.3. Theories and their models . . . . .	152
5.2.3.1. How can we reason in FOL? . . . . .	153
5.3. Semantic tableaux in FOL . . . . .	154
5.4. Unification in the method of semantic tableaux . . . . .	166
5.5. Toward a semi-decision procedure for FOL . . . . .	169
5.5.1. Prenex normal form . . . . .	169
5.5.1.1. Skolemization . . . . .	174
5.5.2. Skolem normal form . . . . .	176
5.6. Semantic trees in FOL . . . . .	186
5.6.1. Skolemization and clausal form . . . . .	188
5.7. The resolution method in FOL . . . . .	190
5.7.1. Variables must be renamed . . . . .	201



5.8. A decidable class: the monadic class . . . . .	202
5.8.1. Some decidable classes . . . . .	205
5.9. Limits: Gödel's (first) incompleteness theorem . . . . .	206
<b>Chapter 6. Foundations of Logic Programming . . . . .</b>	<b>213</b>
6.1. Specifications and programming . . . . .	213
6.2. Toward a logic programming language . . . . .	219
6.3. Logic programming: examples . . . . .	222
6.3.1. Acting on the execution control: cut “/” . . . . .	229
6.3.1.1. Translation of imperative structures . . . . .	231
6.3.2. Negation as failure (NAF) . . . . .	232
6.3.2.1. Some remarks about the strategy used by LP and negation as failure . . . . .	238
6.3.2.2. Can we simply deduce instead of using NAF? . . . . .	239
6.4. Computability and Horn clauses . . . . .	241
<b>Chapter 7. Artificial Intelligence . . . . .</b>	<b>245</b>
7.1. Intelligent systems: AI . . . . .	245
7.2. What approaches to study AI? . . . . .	249
7.3. Toward an operational definition of intelligence . . . . .	249
7.3.1. The imitation game proposed by Turing . . . . .	250
7.4. Can we identify human intelligence with mechanical intelligence? . . . . .	251
7.4.1. Chinese room argument . . . . .	252
7.5. Some history . . . . .	254
7.5.1. Prehistory . . . . .	254
7.5.2. History . . . . .	255
7.6. Some undisputed themes in AI . . . . .	256
<b>Chapter 8. Inference . . . . .</b>	<b>259</b>
8.1. Deductive inference . . . . .	260
8.2. An important concept: clause subsumption . . . . .	266
8.2.1. An important problem . . . . .	268
8.3. Abduction . . . . .	273
8.3.1. Discovery of explanatory theories . . . . .	274
8.3.1.1. Required conditions . . . . .	275
8.4. Inductive inference . . . . .	278
8.4.1. Deductive inference . . . . .	279
8.4.2. Inductive inference . . . . .	280
8.4.3. Hempel's paradox (1945) . . . . .	280
8.5. Generalization: the generation of inductive hypotheses . . . . .	284
8.5.1. Generalization from examples and counter examples . . . . .	288

<b>Chapter 9. Problem Specification in Logical Languages</b> . . . . .	291
9.1. Equality . . . . .	291
9.1.1. When is it used? . . . . .	292
9.1.2. Some questions about equality . . . . .	292
9.1.3. Why is equality needed? . . . . .	293
9.1.4. What is equality? . . . . .	293
9.1.5. How to reason with equality? . . . . .	295
9.1.6. Specification without equality . . . . .	296
9.1.7. Axiomatization of equality . . . . .	297
9.1.8. Adding the definition of = and using the resolution method . . . . .	297
9.1.9. By adding specialized rules to the method of semantic tableaux . . . . .	299
9.1.10. By adding specialized rules to resolution . . . . .	300
9.1.10.1. Paramodulation and demodulation . . . . .	300
9.2. Constraints . . . . .	309
9.3. Second Order Logic (SOL): a few notions . . . . .	319
9.3.1. Syntax and semantics . . . . .	324
9.3.1.1. Vocabulary . . . . .	324
9.3.1.2. Syntax . . . . .	325
9.3.1.3. Semantics . . . . .	325
<b>Chapter 10. Non-classical Logics</b> . . . . .	327
10.1. Many-valued logics . . . . .	327
10.1.1. How to reason with $p$ -valued logics? . . . . .	334
10.1.1.1. Semantic tableaux for $p$ -valued logics . . . . .	334
10.2. Inaccurate concepts: fuzzy logic . . . . .	337
10.2.1. Inference in FL . . . . .	348
10.2.1.1. Syntax . . . . .	349
10.2.1.2. Semantics . . . . .	349
10.2.2. Herbrand's method in FL . . . . .	350
10.2.2.1. Resolution and FL . . . . .	351
10.3. Modal logics . . . . .	353
10.3.1. Toward a semantics . . . . .	355
10.3.1.1. Syntax (language of modal logic) . . . . .	357
10.3.1.2. Semantics . . . . .	358
10.3.2. How to reason with modal logics? . . . . .	360
10.3.2.1. Formal systems approach . . . . .	360
10.3.2.2. Translation approach . . . . .	361
10.4. Some elements of temporal logic . . . . .	371
10.4.1. Temporal operators and semantics . . . . .	374
10.4.1.1. A famous argument . . . . .	375

10.4.2. A temporal logic . . . . .	377
10.4.3. How to reason with temporal logics? . . . . .	378
10.4.3.1. The method of semantic tableaux . . . . .	379
10.4.4. An example of a PL for linear and discrete time: PTL (or PLTL) . . . . .	381
10.4.4.1. Syntax . . . . .	381
10.4.4.2. Semantics . . . . .	382
10.4.4.3. Method of semantic tableaux for PLTL (direct method) . . . . .	383
<b>Chapter 11. Knowledge and Logic: Some Notions . . . . .</b>	<b>385</b>
11.1. What is knowledge? . . . . .	386
11.2. Knowledge and modal logic . . . . .	389
11.2.1. Toward a formalization . . . . .	389
11.2.2. Syntax . . . . .	389
11.2.2.1. What expressive power? An example . . . . .	389
11.2.2.2. Semantics . . . . .	389
11.2.3. New modal operators . . . . .	391
11.2.3.1. Syntax (extension) . . . . .	391
11.2.3.2. Semantics (extension) . . . . .	391
11.2.4. Application examples . . . . .	392
11.2.4.1. Modeling the muddy children puzzle . . . . .	392
11.2.4.2. Corresponding Kripke worlds . . . . .	392
11.2.4.3. Properties of the (formalization chosen for the) knowledge . . . . .	394
<b>Chapter 12. Solutions to the Exercises . . . . .</b>	<b>395</b>
<b>Bibliography . . . . .</b>	<b>515</b>
<b>Index . . . . .</b>	<b>517</b>



## Preface

These notes result from a certain conception of knowledge transmission and from the experience gained after several years of teaching at the Grenoble Institute of Technology (Ensimag).

If the table of contents is interpreted too literally, the task at hand is infeasible: each of the themes developed in the different chapters has been the topic of thousands of pages (books, monographs, articles, popularization books, etc.) published by numerous authors, and some of these pages are of the highest scientific quality. On top of this, we must consider all the information available on the Internet.

The aim of these notes, which is probably ambitious but hopefully not disproportionate, is to attempt to provide a unified overview of the concepts and techniques that are useful in many well-identified domains of modern computer science and artificial intelligence. It is difficult to find all these topics in the same document, and they should also be a good starting point for a reader wishing to explore further topics.

Conceptual rigor will always be preferred to formal rigor. This approach is essential for the transmission of knowledge in the modern world, especially for those domains in which the readers will have to keep acquiring additional knowledge throughout their professional life.

The presentation method of all the topics will always be the same: informal description of the topic under consideration (motivation)  $\rightsquigarrow$  historical background  $\rightsquigarrow$  examples  $\rightsquigarrow$  possible conceptualizations  $\rightsquigarrow$  comparative analysis  $\rightsquigarrow$  formalization  $\rightsquigarrow$  technical aspects.

Of course, the algorithmic point of view is privileged and for almost every considered problem, the goal is to design an algorithm capable of solving it. Examples play a crucial role in these notes: they have been chosen so as to guide in the

conceptualization of pertinent abstractions for classes of problems. Digressions and remarks allow for an in-depth view of some of the topics and for the discovery of their relationship with other topics and other domains. Exercises are an essential complement to the topics treated, which cannot be understood and assimilated without solving them (solutions to the exercises are included in the final chapter). It is clear that the material treated here has already been discussed in other books. However, some of these topics are approached in an original manner in this book.

In addition to carrying out their original goal within a university syllabus, these pages will hopefully be agreeable to the reader and will also be an incentive to those wanting to know more and ask further questions.

## Chapter 1

# Introduction

We briefly analyze the relationship between logic and computer science, by focusing successively on two points of views, in a somewhat natural order. We first underline the importance of logic in the foundations of computer science and how it is used in many computer science domains. Then we explain why logic is useful for computer engineers.

### **1.1. Logic, foundations of computer science, and applications of logic to computer science**

Trying to underline the importance of logic for computer science in the 21st Century is the same as trying to reinvent the wheel. Indeed, in 1969, C.A.R. Hoare wrote:

Computer programming is an exact science, in that all the properties of a program and all the consequences of executing it can, in principle, be found out from the text of the program itself by means of purely deductive reasoning.

More recently, in 1986, Hoare stated that “computers are mathematical machines and computer programs are mathematical expressions”.

Of course, in our defence of logic, we will not make use of arguments relying on Hoare’s renowned expertise, as these arguments may turn out to be fallacious; however, we attempt to shed some light on this kind of sentence and to convince the reader that the importance of logic for computer science is a fact and not a matter of opinion.

## 2 Logic for Computer Science and Artificial Intelligence

Logical concepts have been of prime importance in computer science, and this is still the case nowadays.

Instead of making a potentially tedious enumeration, we shall simply mention two typical concepts of computer science, *production rules* and *formal languages*, the origins of which are seldom mentioned and which were, respectively, invented by logicians in 1921 (Post) and 1879 (Frege).

We cannot overstate the importance of studying the foundations and the history of a discipline, as proved by the following quote.

In 1936, A. Turing introduced his notion of an abstract computer, as part of his solution to one of the problems posed by D. Hilbert. Yet, H. Aiken (one of the pioneers of computer science) wrote in 1956 (as quoted by M. Davis):

If it should turn out that the basic logics of a machine designed for the numerical solution of differential equations coincide with the logics of a machine intended to make bills for a department store, I would regard this as the most amazing coincidence that I have ever encountered.

Such a statement would make any undergraduate-level computer scientist smile today.

Another point of view, which does not have such a good press currently, is the *philosophical* point of view. To convince oneself of its importance, it suffices to recall the role of intuitionisms in computer science (constructive program synthesis, etc.). Several philosophical questions arise naturally in elementary logic (paradox, truth value, possible worlds, intention, extension, etc.).

The importance of temporal logic in modern computer science is undeniable, and is due (including for philosophical motivations) to philosophers–logicians such as A. N. Prior (see section 10.4).

The philosophical point of view is essential to *understand* a topic, and of course, understanding is crucial from a *practical* point of view.

An example is the popular term *ontology*. J. McCarthy borrowed it in the 1970s for philosophy. Currently, in computer science and artificial intelligence (AI) the meaning of this term has connections (even though they may not be that obvious) with its original philosophical meaning, i.e. “Theory of being as such – the branch of metaphysics that deals with the nature of being, as opposed to the study of their memberships or attributes”.

We conclude this section by mentioning three so-called *theoretical* topics, of the utmost *practical* importance:



– The NP-completeness of the consistency problem (satisfiability and validity) of classical propositional calculus (SAT), which was proved by Cook in 1971. The wide array of applications of propositional calculus (verification of critical systems, intelligent systems, robotics, constraints, etc.) provides an idea of the importance of this result.

– The study of *finite* structures, which is closely related to computer science and has numerous applications in databases, multi-agent systems, etc.

– *Non-classical logics* (modal, temporal and multi-valued logics) are extremely useful in, e.g. program analysis and knowledge representation (in particular in distributed knowledge representation).

## 1.2. On the utility of logic for computer engineers

Although no one could say for sure which concepts and techniques will be useful to a computer scientist in, say, 10, 20, 30, or 40 years, or even if computer scientists will still be around by then, human beings will probably not have changed that much and will have to cope with an increasingly complex world (at least in developed countries). Abstract concepts will turn out to be more and more useful from the point of view of *understanding* and *efficiency* (e.g. it suffices to keep in mind all the advantages – conception, encoding, and updating – that offer very high-level languages).

It is also important to remember that one of the most efficient and rewarding recipes to success from an economical point of view, especially in the modern world, is *originality*. History shows that the original ideas that have led to important progress in science and techniques are mostly a consequence of an in-depth analysis (and possibly a revision) of principles.

Some of the fundamental notions of logic are used by computer engineers (sometimes with different names and presentations). For example, proposition, definition, semantics, inference, language/meta-language, intention, intension, extension, subclass of formulas, and model.

The notion of proposition from an intuitionistic point of view, for example, represents an intention, a task or a problem. The notion of definition, which has been studied for centuries by logicians, is closely related to that of specification and is used in the process of program construction (folding and unfolding rules).

The role of logic as a language for software specification/verification/development is unchallenged. The notion of semantics leads directly to that of compilation (the assignment of a semantics can be viewed as a translation). Inference is a way of expliciting information and is therefore strongly related to algorithms design. The notion of intention is related not only to that of specification but also to that of system modeling (and not only program modeling), in which it is also necessary to model the

#### 4 Logic for Computer Science and Artificial Intelligence

environment (including users). The notions of intension and extension are used, for example, in languages such as Datalog (studied in logic and databases). The notion of decidable subclasses of formulas is an example of a problem whose *nature* changes when one of its subproblems is considered. The notion of models in the context of abduction (which is used for example in diagnostics, or in the understanding of natural languages) is similar to that of empirical sciences.

These are, of course, very general concepts, but let us mention some concrete examples that have extremely important practical applications. The Davis–Putnam method is used, for example, in system validation. The notion (algorithm) of subsumption is used, among other things, in knowledge representation languages.

This notion is essential in so-called ontologies (considered as sets of concepts, relationships between these concepts and means to reason on these concepts), which are widely used in taxonomies and sometimes defined as *the explicit specification of a simplified and abstract view of a world we wish to represent*.

The notion (algorithm) of unification has been at the intersection of logic and computer science for many years. This algorithm is specified in a very natural way as a set of inference or rewriting rules (see section 4.2). Databases are one of the traditional areas of application of computer science. We can view a relational database as a first-order structure, and first-order logic as a query language (see remark 6.1). The search for increasingly powerful query languages shows the practical need for logics with a greater expressive power.

These examples should be enough to convince the reader of the importance of logic for computer engineers.

To answer those for whom the incentive to learn logic can only come from its recent applications or its relationship to recent applications, we enumerate some applications that have been receiving increasing attention.

The first application is multi-agent systems, which are used in important domains such as robotics, the Internet, etc. These systems can possess extremely complex configurations, and it is becoming necessary to have (formal) techniques to reason in (and on) these systems. The notions and tools that are used include temporal logic, knowledge representation logics, deduction, abduction (i.e. the discovery of hypotheses that are capable of explaining an observation or permit us to reach a conclusion), and the notion of *proof* that can be used to convince an agent (in particular a human agent) of the pertinence of a piece of information, etc.

Modal logics (temporal logic, dynamic logic, etc.) are used in the industry, for example, to capture reactive behaviors, or in concurrent systems, etc.

An increasing number of computer engineers now work in the economic and financial industry. In these disciplines, the modeling of the actors and their knowledge (beliefs) of the other actors (on the market, in the society) is indispensable for comprehension and decision. Assertions such as “X knows (believes) that Y knows (believes) that Z knows (believes) that...” can be treated formally by knowledge (belief) logics.

In a science popularization article that was published in a well-known journal in May 2001, we could read:

The semantic web is... an extension of the current one.

[...]

For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they use to conduct automated reasoning.

Logic is very important for natural language processing, which has many applications (e.g. on the Internet, for information retrieval, in question answering systems, etc).

Interdisciplinarity has reached the most recent computer science applications, such as multimedia indexing (i.e. the way of finding multimedia documents in digital libraries), where the roles of semantics and inference are essential.

We conclude this section with some considerations that are more directly related to the technical problems that arise in computer system programming.

– It is possible to prove that a program is not correct, but it is generally impossible to prove that a program is correct using examples. The importance of detecting potential errors in programs (or in integrated circuit designs) is evidenced by two dramatic examples that took place 30 years apart.

The spacecraft *Mariner 1* (July 1962) was meant to land on Venus but failed because of an error in the on-board program (a syntactically correct instruction, very close to the intended one, had an entirely different meaning).

Logical techniques have been developed to prove program correctness, to detect programming errors and to construct programs that satisfy a given specification. Most of these techniques can be performed automatically. To reason automatically on a program, it is necessary to have a formal semantics, a formal logical theory, and an automated theorem prover for this theory.

In general, we are interested in verifying specifications, i.e. in proving properties satisfied by the specifications rather than in proving properties satisfied by the programs themselves.

Nowadays, people use more and more *certified software*. The practical importance of this notion cannot be overstated (it suffices to reflect on the importance of having certified software in a plane or a hospital). More recently (mid-1990's), errors were discovered in a digital circuit that had already been commercialized. It had been sold by the biggest manufacturer of digital circuits. This error occurred when rare data were provided to the circuit.

This error served as an argument to those who advocated for the necessity of replacing simulation with exhaustive tests by formal verification, so as to prove the correctness of a design before the construction phase.

*Theorem proving* and *model checking* are two techniques that enable us to certify the correctness of the aforementioned digital circuits. Logic (both classical and non-classical) plays a key role in these two approaches.

Some resounding successes in software and hardware engineering have proved that these approaches can indeed be applied to real-world problems.

Formal verification had typically been a neglected domain because it was considered too theoretical, but now, it can have a huge financial impact.

– Logic programming consists of using logic as a programming language. The programming language Prolog, as well as others that followed, in particular, constraint logic programming languages, arose naturally from the notions and methods that are studied in logic.

– Some logical paradoxes, in particular, the one named Russell's paradox, are closely related to the halting problem, i.e. the existence of an algorithm that can decide whether any program provided as an input will halt or not. The impossibility of creating such an algorithm is well known.

– Over the past few years, there has been a boom of computer systems that exhibit an intelligent behavior (such systems are mostly studied in the discipline known as AI).

The principles that are used to design these systems are closely related to classical and non-classical logic. The role of logic in social sciences must not be discarded. Logic is considered as a preferred tool in, e.g. the study of intelligent interactions.

## Chapter 2

# A Few Thoughts Before the Formalization

### 2.1. What is logic?

We cannot give a formal answer to this question right away (we will get back to it though). In order to be understood, the answer would require some hindsight on the topic about to be studied.<sup>1</sup>

Try to understand what mathematics is about by only relying on its definition: mathematics, the science of quantity and space.

To choose an answer would not be very helpful right now, because a lack of criteria and references to concrete cases make it difficult to judge the pertinence of the answer.

A problem that is closely related to the one under consideration inspired the following thought to a famous philosopher (D. Hume):

The fact that ideas should logically entail one another is nothing but a fact, no more understandable by itself than any fact of the material world.

And this one from another less famous philosopher:

A logical formula is the expression of a natural phenomenon, as is gravitation or the growth of a tree.

---

<sup>1</sup> This remark is valid for any topic about to be studied.

### 2.1.1. *Logic and paradoxes*

Let us return to the history of logic and try to analyze some of the well-known concepts that are involved.

Logical difficulties arose very early in philosophy, religious “treaties”, and literature. Here are two examples:

- the liar’s paradox, due to Eubulides of Miletus (see digression 2.2): I am lying;
- the sentence: **this sentence is false**;
- the version of the liar’s paradox due to Epimenides of Knossos (6th Century BC): All Cretans are liars.

Are these really paradoxes?

What is a paradox?

Etymologically (15th Century AD): **paradox**: contrary to common opinion (doxa: opinion, from which originated *heterodox* and *paradox*).

Other definitions characterize paradoxes as argumentations (or assertions) that lead to a contradiction (logical paradoxes).

Paradoxes are sometimes associated with results (obtained by using correct reasoning) that are contrary to intuition and common sense, thus provoking surprise and perplexity.

One may also call a paradox a proposition that seems true (false) and is actually false (true).

Consider the following story (an excerpt from a masterpiece of universal literature):

A soldier is given the order to ask every person about to cross a bridge the following question:

(\*) “What have you come here for?”

- If the person tells the truth, he is allowed to cross the bridge.
- If the person is lying, he must be hanged next to the bridge.

Someone arrives, and when asked question (\*), shows the gallows next to the bridge and replies: “I have come to be hanged in these gallows”.

Imagine how embarrassed the soldier must feel, as he must either allow someone who lied to cross the bridge, or hang someone who was telling the truth!

There also exist paradoxes that occur in board games.

The rule **Every rule has exceptions** gives rise to problems. As it is a rule, it must admit some exceptions. Which means that there exist rules that do not admit any exception.

Here is another one:

i) **The sentence below is false.**

ii) **The sentence above is true.**

If (i) is **T** then (ii) is **F** hence (i) is **F**.

If (i) is **F** then (ii) is **T** hence (i) is **T**.

If (ii) is **F** then (i) is **T** hence (ii) is **T**.

If (ii) is **T** then (i) is **F** hence (ii) is **F**.

### 2.1.2. *Paradoxes and set theory*

Perhaps those paradoxes that had the most impact are those that involve set theory: probably because of the importance of set theory in mathematics, and particularly in the foundations of mathematics.

Bolzano introduced (in 1847) the notion of a “set” for the first time:

*A set is a collection of elements the order of which is not pertinent, and nothing essential is changed by only changing this order.*

But it is Cantor who developed set theory.

In naive set theory, according to Cantor, a set is:

*Any collection of objects from our intuition or our thoughts, that are both defined and different.*

... But allowing the use of *any property* to define a set can be dangerous:

There exist sets that do not contain themselves. For example:

*The set of prime numbers < 150*

There are others that contain themselves. For example:

The set of all ideas.

The catalog of all catalogs.

From a more abstract point of view:

The set of all sets.

It is simple to show that accepting as a set all the sets in the universe ( $U$ ) leads to a problem.

If this is a set, it must contain itself. But the set of its subsets  $\mathcal{P}(U)$  is also a set, and  $\text{card}(\mathcal{P}(U)) > \text{card}(U)$ . Thus, there would exist a set containing more sets than the universe itself!

The set of all the sets that can be described in English by fewer than twenty words.

Bertrand Russell came up in 1902 with a “set” that leads to a paradox, which is known as “Russell’s paradox”:

Let  $B$  denote the set of all sets  $A$  such that  $A$  is not an element of  $A$  (i.e. the set of all sets that do not contain themselves).

The existence of  $B$  leads to a contradiction:

– If  $B \in B$ , then  $B$  contains a set that contains itself ( $B$ ), and  $B$  must not contain  $B$  (as  $B$  only contains those sets that do not contain themselves), hence  $B \notin B$ .

– If  $B \notin B$ , then as  $B$  does not contain itself,  $B$  must contain  $B$  (as  $B$  contains all the sets that do not contain themselves), hence  $B \in B$ .

Russell’s paradox was popularized as the “barber’s paradox”. A barber must shave every man who does not shave himself. How about him? Must he shave himself? If he does, then he is shaving someone who is shaving himself. Thus, he must not shave himself. If he does not shave himself, then he will not be shaving every man who does not shave himself. Hence, he must shave himself.

The origin of this problem is the axiom of abstraction (also called the axiom of naive comprehension): *Given a property, there exists a set whose members are exactly the entities satisfying this property.* In other words, *for every predicate  $P$ , there exists a set whose elements are all the objects (and no other) that satisfy  $P$ :*

$$\exists x \forall y (y \in x \Leftrightarrow P(y))$$

#### 2.1.2.1. The answer

The paradoxes on set theory were believed to be caused by the definition of incomplete objects as sets. Such objects should not be considered as sets.

When a set  $S$  and an object  $ob$  are defined in such a way that:



- i)  $ob$  is an element of  $S$ ;
- ii) the definition of  $ob$  depends on  $S$ ,

the definition is said to be *impredicative* (Poincaré). This is the same as what is commonly called a *vicious circle*.

To avoid paradoxes, impredicative definitions must be considered as illegitimate.

In set theory, this leads to the distinction between a *class* and a *proper class*.

A *class*:  $\{x \mid P(x)\}$  ( $P$  predicate symbol).

Sets are *classes*, but not all classes are sets:

$\{x \mid x \in x\}$  (or  $\{x \mid x \notin x\}$ ) is not a set but it is a *proper class*.

Sets are complete entities.

Classes are incomplete entities.

In set theory, a constructive definition of a hierarchy of sets is provided (as usual,  $\mathcal{P}(X)$  represents the set of all the subsets of a set  $X$ ):

$$\mathcal{F}_0 = \emptyset$$

$$\mathcal{F}_{n+1} = \mathcal{P}(\mathcal{F}_n)$$

$$\mathcal{F}_\omega = \cup_i \mathcal{F}_i$$

It is worth mentioning that impredicative but still important definitions are used in mathematics, such as the definition of the *least upper bound* (it is among all the bounds under consideration), *maximum of a function on an interval* (the maximum is among all the considered values), etc.

These kinds of definitions have also proved their value in computer science (communicating systems and *streams*).

*A set belongs to a family of sets.*

The *axiom of choice* (AC) and the *continuum hypothesis* (CH) are of great importance in set theory and in the foundations of mathematics.

We present three versions of the AC:

**Version 1:**

For every set  $X$  of non-empty sets  $y$ , there exists a function  $f$  with domain  $X$ , such that  $f(y) \in y$  for all  $y \in X$ . (It is necessary to use this version only when  $X$  is infinite).

**Version 2:**

Let  $S$  denote a set of paired disjointed, non-empty sets. There exists a set  $C$  containing exactly one element in each member of  $S$ .

**Version 3 (formalization of version 1):**

$$\forall X \exists f [f \text{ a function with domain } X \wedge \forall z ( \underbrace{z \in X \wedge \exists u . u \in z}_{X: \text{set of non-empty sets}} \Rightarrow f(z) \in z )]$$

The CH states that between  $\aleph_0$  (the cardinality of  $\mathbb{N}$ ) and  $\aleph_1$  (the cardinality of  $\mathbb{R}$ , also denoted by  $2^{\aleph_0}$ ), there is no other transfinite cardinal.

The generalized CH states that the sequence of transfinite cardinals is:

$$\aleph_0, \aleph_1 = 2^{\aleph_0}, \aleph_2 = 2^{\aleph_1}, \dots, \aleph_{n+1} = 2^{\aleph_n}, \dots$$

The *method* used to construct this sequence is to consider a set and the set of its subsets. For  $\aleph_0$  ( $\text{card}\mathbb{N}$ ), the method is the one shown in exercise 3.1.

For the others, assume that there exists a bijection  $f$  between  $E$  and  $\mathcal{P}(E)$ . By diagonalization (as in exercise 3.1), we can prove that  $f$  cannot be onto.

Gödel proved (in 1940) that if the ZF (Zermelo–Fraenkel) axiomatization is consistent (i.e. does not permit us to deduce a formula and its negation), then ZF + AC and ZF + AC + HC are also consistent. This means that AC and HC cannot be refuted.

Paul Cohen proved (in 1963) that AC and HC are *independent* from the other axioms in ZF, which means that AC and HC cannot be proved if ZF is consistent. HC cannot be proved in ZF + AC.

*ZF and AC are undecidable in ZF.* □

**DIGRESSION 2.1.**– (sets and the AC in constructivism). For intuitionists or constructivists, or at least for most of them, a set  $E$  is well defined if and only if:

- i) we are told how to construct an element of  $E$ ;
- ii) we are told how to prove that two elements in  $E$  are equal;
- iii) we are given a proof that the equality defined in (ii) is an equivalence relation.

If only item (i) is satisfied,  $E$  is called a pre-set (it is possible to define sets, using pre-sets as a starting point).

A set  $E$  is completely presented iff for any of its elements, one can “read” evidence that it belongs to  $E$  (for example,  $\mathbb{Q}$  is completely presented, as for every  $\frac{m}{n}$ , it is possible to verify that  $m$  and  $n$  have no common factor).

Constructivists consider the AC as *essentially non-constructive*; however, they accept some (constructive) versions that are *provable* in constructive mathematics (the name “axiom” was kept out of respect for tradition in classical mathematics). We provide the version named denumerable choice:

$$\mathbf{AC}_{\mathbb{N}}: E \subset \mathbb{N}^2 \wedge \forall m \exists n (< m, n > \in E) \Rightarrow \exists f: \mathbb{N} \rightarrow \mathbb{N} \mid \forall m (< m, f(m) > \in E)$$

□

### 2.1.3. Paradoxes in arithmetic and set theory

EXAMPLE 2.1.– (Berry’s paradox). Consider the formalization of arithmetic based on Peano’s axioms (see example 3.8). □

Let  $\mathcal{A}$  denote the set of all natural numbers that can be defined in English by a sentence containing at most a thousand characters (letters or separation symbols).  $\mathcal{A}$  is therefore finite, and there must exist natural numbers that do not belong to  $\mathcal{A}$ . The sentence:

$n$  is the smallest integer that cannot be defined by an English sentence containing at most a thousand characters

contains fewer than a thousand characters and defines a natural number  $n$ . Hence,  $n$  is a member of  $\mathcal{A}$ . However, by definition,  $n$  does not belong to  $\mathcal{A}$ .

We shall soon return to the notions of theory and meta-theory.

#### 2.1.3.1. The halting problem

The idea underlying Russell’s paradox can be used to prove the undecidability (insolubility) of the *halting problem*. This idea can be characterized as the possibility for an expression designating an object to refer to a whole that the object belongs to.

The problem can be stated as follows:

Halting problem: given a program in a language (with the same expressive power as the Turing machine)<sup>2</sup> and some input data<sup>3</sup> (or an initial state of its input tape), decide whether this program (or Turing machine) halts or not.

To prove that this problem can be solved for a certain language, it suffices to provide a program that solves it.

---

<sup>2</sup> Of course, leaving this language unspecified does not entail any loss of generality in the statement.

<sup>3</sup> This can be any input data.

```

program D;
begin
  while A(D)
  do
     $x \leftarrow x \%$ , i.e. do not do anything
  end while
  return T
end

```

**Figure 2.1.** *The halting problem is unsolvable*

However, to prove that it cannot be solved, it is necessary to use an *indirect* method (for example *reductio ad absurdum*).

It is possible to encode a program as a set of input data (see, e.g. section 5.9).

We therefore *assume* that we can write a program *A*, which decides for *any* program *P* whether *P* halts or not, i.e.  $A(P)$  returns `true` if and only if *P* halts.

We define the program *D* of Figure 2.1.

- Program *D* halts if *D* does not halt;
- Program *D* does not halt if *D* halts.

*Contradiction.* Hence *A* cannot exist. □

**DIGRESSION 2.2.**– (on the importance of paradoxes). Paradoxes can be viewed as questions about the foundations of our reasonings, of our definitions, of our levels of language, etc. They have greatly influenced traditional logic, and also modern logic, set theory, etc. The principles of paradoxes can be used to obtain fundamental results in mathematical logic (for example Berry’s paradox and the proof of Gödel’s incompleteness theorem, see section 5.9) or in computability (the halting problem).

A typical historical figure that came up with several paradoxes is Eubulides of Miletus (circa 384–322 BC), who was contemporary with Aristotle, with whom he was at odds and against whom he produced several writings.

He is believed to be the father of *eristic* arguments (i.e. related to controversies, specious reasoning sophisticated quibbling).

It seems like the refutation of these arguments played a central role in the design of Aristotelian logic.

He is the one who came up with the liar's paradox<sup>4</sup>, and that of the heap of sand (see example 8.1). The latter poses the problem of the relationship between the continuum and the discrete.

There also exist paradoxes (that are not necessarily logical paradoxes but do illustrate the etymology of the word "paradox") in other scientific domains.

For example:

The paradox known as the *Tocqueville effect*: revolutions always start when a totalitarian regime becomes more liberal.

The *Einstein–Podolsky–Rosen* paradox involving quantum physics.

*Bertrand's* paradox: one has to determine the probability that a randomly chosen chord in a circle is longer than the side of the equilateral triangle that admits the same circle as a circumcircle. Using the definition of Laplace (probability = number of favorable cases ÷ total number of cases) leads to two different values depending on the way one chooses to define the favorable cases. And both choices are equally reasonable. □

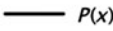


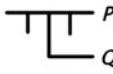
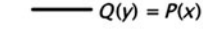


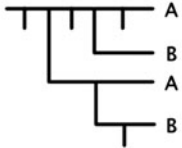
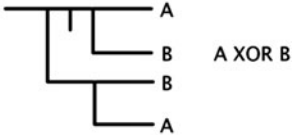
#### 2.1.4. *On formalisms and well-known notions*

G. Frege (19th to 20th Century) was one of the greatest logicians in history. He introduced the notions of *formal systems*, *quantifiers*, and *proofs* that are still used today.

The formalism he proposed was too cumbersome, as evidenced by the following examples.

---

<sup>4</sup> Legend has it that the poet and philosopher Philotas of Cos was subject to a deteriorating health as he tried to analyze this paradox.

Frege's notation	Modern notation
 $P(x)$	$P(x)$
 $P(x)$	$\neg P(x)$
 $P(x)$ $Q(y)$	$Q(y) \Rightarrow P(x)$
 $P(x)$ $Q(y)$	$\neg (Q(y) \Rightarrow \neg P(x)) : Q(y) \wedge P(x)$
 $Q(y) = P(x)$	$Q(y) \Leftrightarrow P(x)$
 $P(x)$	$\forall x P(x)$
 $P(x)$	$\neg \forall x \neg P(x) : \exists x P(x)$
 A B A B	OR
 A B B A	A XOR B

This formalism was not used very much afterwards (which is not surprising!)<sup>5</sup>.

We shall often ask ourselves questions about notions that seem natural. To convince ourselves that important problems are often hidden by our habits, it is enlightening to ponder, for example, on a word that can often be found in mathematical statements:

Does there exist... such that...?

- 1) According to formalists, **existence** means consistency, non-contradiction.
- 2) According to intuitionists<sup>6</sup>, **existence** means construction, meaning.

<sup>5</sup> One cannot overstate the importance of formalism. For example, McColl proposed  $P : Q$  as a notation for the implication  $P$  implies  $Q$ . But the symmetry in this notation suggests that  $P$  and  $Q$  have the same role... it is thus a bad formalism.

<sup>6</sup> Some authors prefer to use the term *constructivists* (and *constructivism*), which they believe better mirrors the underlying philosophy. Brouwer, one of the founders of constructivism, viewed mathematics as the activity of carrying out constructions (a primitive and somewhat vague concept) in the mind of an ideal mathematician.

In mathematics that almost everyone uses, it is the first point of view that is more or less consciously adopted, although this notion may not be as natural as it may seem:

The following example, proposed by Frege, should make us think.

Consider the three following propositions:

- G is an intelligent being;
- G is ubiquitous;
- G is all-knowing,

and suppose that these three propositions as well as all their consequences are non-contradictory.

*Can we conclude that G exists?*

EXAMPLE 2.2.– Do there exist irrational numbers  $a, b$  such that  $a^b$  are rational? *Classical* response: yes. *Classical* proof:

Take  $a = \sqrt{2}$  and  $b = \sqrt{2}$ , then:

$$\sqrt{2}^{\sqrt{2}} \begin{cases} \text{rational} & \text{result proved} \\ \text{irrational} & \text{take } a = \sqrt{2}^{\sqrt{2}} \text{ and } b = \sqrt{2} \end{cases}$$

(in the second case,  $a^b = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^2 = 2$ ).

This is not a constructive proof because no one can say which of the two possibilities proves the conjecture.

Here is a constructive proof

$$\text{constructive proof} \begin{cases} a = \sqrt{2} & b = 2 \log_2 5 \\ a^b = \sqrt{2}^{2 \log_2 5} = 2^{\log_2 5} \stackrel{\text{def. log}}{=} 5 \end{cases}$$

Note that, from a standard intuitionistic point of view, it is permitted to state in a proof that, for example,  $10^{10^{10}} + 1$  is a prime number or is not a prime number. This is because there exists an *effective method* that allows us to decide which of the two hypotheses is correct: it suffices to try all potential divisors.  $\square$

A (primitive) key concept of intuitionism is that of *constructions*. This concept can sometimes shake our beliefs about our understanding of mathematics.

The notion of construction appears to be inherent to every domain of study, and it may not be obvious to tell whether a proof is constructive. For example, consider the way Euclid proved that there exists an infinite number of prime numbers:

Assume that there only exists a finite number of prime numbers, and put these numbers in increasing order

$$p_1, p_2, \dots, p_n$$

$p_n$  is thus the greatest prime number. But in this case,  $p_1 \times p_2 \times \dots \times p_n + 1$  is also a prime number that is greater than  $p_n$ . This is a contradiction, hence there are infinitely many prime numbers.

This proof method uses the law of excluded middle, but it *also* provides a way of *constructing* an infinity of prime numbers...

The following example, which was proposed by Brouwer, one of the founders of intuitionism, clearly shows the importance this school of thought attaches to *what is known* and not just to *what is*.

EXAMPLE 2.3.— Let  $\alpha$  denote the position in the development of  $\pi$ , where the sequence 0123456789 occurs for the first time, and let  $n$  denote the integer defined as follows:

$$n = \begin{cases} 1 + (-1)^\alpha / 10^\alpha & \text{if } \alpha \text{ exists} \\ 1 & \text{otherwise} \end{cases}$$

If  $\alpha \in \mathbb{N}$  and  $\alpha$  is even, then:

$$n = 1. \underbrace{00 \dots 01}_\alpha = 1 + 1/10^\alpha$$

If  $\alpha \in \mathbb{N}$  and  $\alpha$  is odd, then

$$n = 0. \underbrace{99 \dots 99}_\alpha = 1 - 1/10^\alpha$$

However, we (still) do not know whether  $\alpha$  exists or not; we therefore have construction rules that we do not know how to apply. This definition only produces an approximation of  $n$ .  $\square$

To quote the mathematician H. Weyl (another important figure of intuitionism):

To provide an existence proof that is not constructive is like announcing that there is a treasure hidden somewhere, without saying where it is exactly.



For intuitionists, the sentence  $A \vee \neg A$  is not valid for all assertions  $A$ . This does not mean that  $A \vee \neg A$  is false, as that would mean that a contradiction could be deduced from the sentence. Let  $A$  denote some mathematical conjecture. As long as  $A$  remains a conjecture, no one can state  $A$ , but as one cannot deduce a contradiction from  $A$  either, no one can state  $\neg A$ .

#### 2.1.4.1. Some “well-known” notions that could turn out to be difficult to analyze

##### a) Definitions and principle of the law of excluded middle

We are used to employing the law of excluded middle when carrying out a proof, especially for proofs involving *reductio ad absurdum*.

Yet there are some cases in which this use is problematic, especially from the point of view of a computer scientist or simply someone who is not content knowing that an object can be defined (this information by itself may not be that helpful), but also wants to know how this object can be *constructed*.

The following example should ring a bell to computer scientists, as it is related to the halting problem (or the impossibility of enumerating recursive functions).

EXAMPLE 2.4.– Consider a sequence of integers  $a_{mn}$ , where  $m, n \in \mathbb{Z}$ .

Let  $f$  denote the following function:

$$f(m) = \begin{cases} 0 & \text{if } \forall n \ a_{mn} = 0 \\ 1 & \text{if } \exists n \ / \ a_{mn} \neq 0 \end{cases} \quad \square$$

For a classical mathematician, this function is well defined, as for any given  $m$  (using the law of excluded middle), either  $a_{mn} = 0$  for all  $n$ , in which case  $f(m) = 0$ , or there exists an  $n$  such that  $a_{mn} \neq 0$ , in which case  $f(m) = 1$ .

But what if we want to *actually compute* the values of  $f(m)$ ? Then it is necessary to examine an infinite number of terms:  $a_{mn_1}, a_{mn_2}, a_{mn_3} \dots$ , which is of course *impossible*.

##### b) The notion of a proof

In classical logic, the meaning of the connectives  $\wedge$ ,  $\vee$  and  $\Rightarrow$  is given by a combination of the truth values of the sub-formulas that are combined by these connectives.

In intuitionistic logic, their meaning is a result of what should be considered as a proof of the formula containing these connectives. Here, a “proof” should be understood as *a proof using correct means*, and not necessarily a proof in a given formal system (see section 3.3 and remark 5.29 ).

- $p$  proves  $A \wedge B$  iff  $p$  is a pair  $\langle r, s \rangle$ ,  
 $r$  proves  $A$  and  $s$  proves  $B$ .
- $p$  proves  $A \vee B$  iff  $p$  is a pair  $\langle n, r \rangle$ :  
 if  $n = 0$  then  $r$  proves  $A$ ; if  $n = 1$  then  $r$  proves  $B$ , which means that we are given  
 a proof of  $A$  or of  $B$ , but we cannot tell which one it is<sup>7</sup>;
- $p$  proves  $\neg A$  iff  $p$  proves  $A \Rightarrow \perp$   
 %  $\perp$ : contradiction (this symbol is sometimes used to denote something undefined  
 or false);
- $p$  proves  $\perp$ : impossible;
- $p$  proves  $A \Rightarrow B$  iff  $p$  is a rule  $q$  that transforms every proof  $s$  of  $A$  into a proof  
 $q(s)$  of  $B$  (together with any additional information that serves to convince that this is  
 the case)<sup>8</sup>.

First-order logic is introduced in Chapter 5, but for the sake of homogeneity, we introduce the rules corresponding to  $\exists$  and  $\forall$ .

– A set  $X$  is *completely presented* if for all  $x \in X$ , one can examine evidence that  $x$  is indeed in  $X$  (see digression 2.1).

–  $p$  proves  $\exists x \in X \varphi(x)$  iff  $p$  is a pair  $\langle x, q \rangle$ , where  $x$  is a completely presented member of  $X$  and  $q$  is a proof of  $\varphi(x)$  (in other words, we explain *how* to obtain  $x$ ).

–  $p$  proves  $\forall x \in X \varphi(x)$  iff  $p$  is a rule  $q$  such that, for all  $x \in X$  that is completely presented,  $q(x)$  is a proof of  $\varphi(x)$  (together with any additional information that serves to convince that this is the case).  $\varphi(x)$  denotes a predicate on  $X$ , i.e. a rule that assigns to every element  $x \in X$  that is completely presented, a well-formed formula (wff)  $\varphi(x)$ .

REMARK 2.1.– (on intuitionistic logic). Intuitionistic logic is *weaker* (it admits less principles), but its theorems are more *general* (it requires less conditions).  $\square$

### c) Implication

Some of the choices that were made in classical logic sometimes lead to seemingly surprising results.

One such choice is material implication (as usual, **T** represents **true** and **F** represents **false**):

<sup>7</sup> Strictly speaking, this condition is not necessary if the standard notion of a proof is used, as the problem “ $p$  is a proof of  $A$  (respectively a proof of  $B$ )” is decidable.

<sup>8</sup> Some authors point out that this notion of a proof prevents the use of *modus ponens* (see section 3.3.2), when  $A \Rightarrow B$  is one of the premises. Indeed, we would otherwise have an effective method for transforming a proof of  $A$  into a proof of  $B$ : it would suffice to apply *modus ponens*.

$p$	$q$	$p \Rightarrow q$	
<b>T</b>	<b>T</b>	<b>T</b>	
<b>T</b>	<b>F</b>	<b>F</b>	where by definition, $p \Rightarrow q : \neg(p \wedge \neg q)$
<b>F</b>	<b>T</b>	<b>T</b>	
<b>F</b>	<b>F</b>	<b>T</b>	

By randomly selecting two propositions  $p$  and  $q$ , the formula:

$$(p \Rightarrow q) \vee (q \Rightarrow p)$$

is a tautology (i.e. a formula that is always **T**).

Similarly,  $p \Rightarrow (q \Rightarrow p)$  is also a tautology.

It becomes obvious that the definition of  $\Rightarrow$  is too large.

Another example that may throw a beginner off is that:

$$(p \Rightarrow q) \wedge (p \Rightarrow \neg q)$$

is not a contradictory formula (it suffices to choose  $p$ : **F** and  $q$ : **F**). The reason one may think it is contradictory is that, in mathematics, when considering a theorem, one is only interested in the case in which premises are **T**.

The implication connective  $\Rightarrow$  has sometimes been criticized by logic philosophers as not being the adequate translation of *if ... , then ...* in natural language.

In natural language, *if ... , then ...* is used to express, e.g.

- causality: *if* we put a kettle with water in it on top of a fire, *then* the water within will be heated;
- implication: *if* John is smaller than Peter and Peter is smaller than Mark, *then* John is smaller than Mark;
- (possible) explanation: *if* the flowers are on the floor, *then* someone must have dropped them.

On the topic of causality, it is important to note that material implication does not translate this notion adequately, at least for the three following reasons:

1)  $p \Rightarrow p$  is a tautology, whereas causal connection is essentially irreflexive (this means no phenomenon is a cause of itself).

2) A causal relation is essentially asymmetric (non-reversibility), which is not necessarily the case of  $\Rightarrow$ :

$p \Rightarrow q$  and  $q \Rightarrow p$  could both be true.

3)  $p \Rightarrow q$  is **T** if  $p$  is **F** and  $q$  is **T**. If  $p$  is the cause and  $q$  the effect, material implication would translate into *the absence of a cause implies any effect*.

d) *Translations in the language of logic*

The word “and” is commutative, but is it always used in a commutative way?

EXAMPLE 2.5.–

*I was hungry and I went to the restaurant.*

*I was ill and I went to see a doctor.* □

DEFINITION 2.1.– *A unary predicate represents a property (small, mortal, etc). This property is the intension of the predicate.*

*The intension of an  $n$ -ary predicate ( $n > 1$ ) is the relation represented by the predicate.*

DEFINITION 2.2.– *The extension of a unary predicate is the class of individuals that satisfy the property represented by the predicate. The extension of an  $n$ -ary predicate ( $n > 1$ ) is the set of  $n$ -tuples that satisfy the predicate.*

REMARK 2.2.–

*Usual mathematics puts the emphasis on extensionality.*

*Predicates with different intensions may admit the same extension.* □

We refer the reader to section 10.3 for an application of the concepts of intension and extension to computer science. The reader may also see examples 3.8 and 9.28.

DEFINITION 2.3.– *A language is extensional iff any sentence in this language has the same truth value when one of its components is replaced by another component with the same truth value.*

*A connective is extensional if the truth value of composed propositions only depends on the truth value of their components.*

Only using an extensional approach leads to a loss in the expressive power of the language.

EXAMPLE 2.6.– (a non-extensional language (R. Carnap)).

1) *It is raining* **F**    % *it is not raining right now*

2) *It is raining and it is not raining* **F**

3) *It is impossible for there to be*  $\underbrace{\text{rain and no rain}}_{\mathbf{F}}$   $\mathbf{T}$

According to (2), the proposition in (3) (above the brace) has the truth value  $\mathbf{F}$ .

Now, if we proceed as we are used to, and replace this proposition with another that has the same truth value (1), we obtain:

3') *It is impossible for there to be rain*

This proposition obviously has truth value  $\mathbf{F}$ .

Thus, the truth value of a composed sentence was changed by a replacement that does not pose any problem from the extensional point of view.  $\square$

### 2.1.5. *Back to the definition of logic*

We return to the problem that we mentioned at the beginning of this section and enumerate some definitions of logic, knowing fully well that, as a great logician/computer scientist (D. Scott) once said:

(\*) *You have to know some logic to do some logic.*

This is almost the same phenomenon as takes place when learning a natural language: there seems to be an innate underlying structure that limits the syntactic possibilities. The relationship between logic and biology was recently expressed by researchers in cognitive psychology in an article:

(If) logic is indeed the optimal form of biological adaptation...

Requirement (\*) stated above may seem to be circular, but is a means of getting rid of an old paradox that has been tackled by philosophers (in particular by Plato) and that involves knowledge in general: we can neither search for what we know nor for what we do not know, because there is no need to search for what we already know, and what we do not know cannot be searched for, as we do not know what to search for.

This is what a contemporary philosopher calls the “that on which” as a region of pre-comprehension (prior discussion), as opposed to “what”, which is unknown.

In layman’s terms, one could say that principles cannot be postponed indefinitely, or that we cannot try to explain everything about a studied topic before studying it without risking an intellectual paralysis.

#### 2.1.5.1. *Some definitions of logic for all*

Actual way of reasoning, in accordance or not with formal logic.

Abstract and schematic reasoning, often opposed to the complexity of the real world.

Consistent sequence of ideas, way of reasoning correctly, focus.

Reasoning: sequence of propositions related to one another according to determined principles and leading to a conclusion.

Etymologically:

– logic (13th Century): from the Greek word for “the art of reasoning”;

– reasoning: reason  $\longrightarrow$  to count  $\longrightarrow$  computation  $\longrightarrow$  ability to compute or reason.

#### 2.1.5.2. *A few more technical definitions*

Logic studies what ensues from what.

Logic studies consistent sets of opinions (or hypotheses).

Logic is a science of truths.

Logic is a science of deductions.

Logic is a means to evaluate an argument<sup>9</sup>.

Logic studies the methods and principles that are used to distinguish between correct and incorrect reasoning.

At this point, an important question (which is mainly investigated by logic philosophers and psychologists) comes up naturally: *why do we reason incorrectly?* or *why do we make mistakes when reasoning?* This question is closely related to such epistemic questions as: *what does it mean to reason correctly?*, *is the process of reasoning related to constraints from the physical world or to innate human characteristics (or to both)?*, *are the laws of logic revisable?*, etc.

It has been noticed that most of the time, the mistakes humans make while reasoning are not logical mistakes, but those that result from a failure to use some

---

<sup>9</sup> An argument can be defined as a set of propositions, one of which is the *conclusion* while the others are *premises*.

of the premises, or their usage with a different meaning from the one they have in the problem statement, where the immediate perception overshadows the actual relation (i.e. the one that is of interest) between the objects under consideration.

A conceptual and technical remark. The notion of *consequence* has been widely studied from a formal point of view, contrary to that of *non-consequence*, which is as important<sup>10</sup> (see section 8.3.1).

This notion is essential, e.g. for the identification of fallacious reasonings (some authors call these reasonings *fallacies*).

The area of study of formal logic is the analysis of sentences and proofs (the meaning of these words shall be made more precise later on, but their intuitive meaning is sufficient for the time being) by considering their *form* and by abstracting away their content.

We say formal logic (or mathematics) in contrast with philosophical logic (see section 2.2).

- Logic deals with all possibilities and views all possibilities as facts.
- The logic of certainty defines the domain of what is possible.
- A logic shall be designed to capture all *patterns* of reasoning that are meant to be characterized (this approach is necessarily an informal one).

A logic is a collection of mathematical structures, a collection of formal expressions and a satisfactory relation between these two collections.

Something particularly interesting to a computer scientist is the fact that in the past few years, a point of view on logic has steadily been imposing itself:

- modern logic is the foundational discipline for information science and it must be studied from the point of views of information, information processing, and information communication. It includes the study of inference, of computability, etc. Its different aspects are of a particular interest to mathematics, computer science and philosophy, and it is inherent to most of the subjects studied in AI.

This school of thought considers:

- inference as an activity that tries to use facts from the physical world to extract implicit information.

---

<sup>10</sup> As Sherlock Holmes would say: “When you have eliminated what is impossible, what remains must be the truth, no matter how improbable”.

In the tree-like classification of the *Mathematical Review*, “logic and foundations” is at the same level as “set theory” and “number theory”.

Set theory is precisely one of the fundamental themes, and it is what one could call an “empirical” fact that all known mathematics can be based on it (for example with the so-called ZFC formalization, i.e. Zermelo-Fraenkel + axiom of choice).

Formal logic is necessary (at least) to rigorously *state the axioms* and to define the notion of *proof*.

On the topic of deductive reasoning, it is a common remark that one of the characteristics of practical or human reasoning is that the agent freely uses all available knowledge, whereas formal reasoning studies the (reliable) methods that verify that the conclusions are consequences of the premises (“logic is the study of what ensues from what”). Most logicians (as well as philosophers and cognitive psychologists) took an interest in the relationships between both ways of reasoning. It suffices to recall that Gentzen designed his natural deduction system so as “*to set up a formal system that came as close as possible to actual reasoning*”.

The fact that classical logic came short of modeling many characteristics of human reasoning, such as deficiencies of material implication, non-monotony, limited resources, failure to take time into account, etc., led from a theoretical and practical point of view to many very important developments, among which is non-classical logics.

One should finally notice that logic, as is the case for all other topics in science, *changes* with time in its fundamental principles, which can be interpreted differently as time passes by, in its methods, notations, topics of interest, etc.

DIGRESSION 2.3.– (formal definition of logic). We have given a few informal definitions of logic. They were enough to introduce the topic, but the reader who requires rigorous definitions may already be wondering: “but what is the *formal* definition of logic (*abstract*, i.e. independent from the way deductions are carried out)?”, and once more is known on the matter, the same reader will probably wonder “why are there so many logics?” □

There is no formal answer to the second question, but a hopefully satisfactory answer might be that a given logic can be chosen because it is better adapted than another one, for example because it has more expressive power (it allows us to express more things), or because it allows us to say things more naturally, or because it is decidable, or easier to automate, etc.

We have answered the first question. The unifying notions are those of consequence relation or satisfaction (or satisfiability).



Here we introduce the notion of consequence relation. Returning to this definition after having worked on different logics will probably make it easier to grasp.

Algebraic methods play an important role in the definition of logic from an abstract point of view (they had already played an important role in the pioneering work of G. Boole).

DEFINITION 2.4.– (consequence relation (Tarski)). *Let  $\mathcal{L}$  denote a formal language.  $\mathcal{C}_n \subseteq \mathcal{P}(\mathcal{L}) \times \mathcal{P}(\mathcal{L})$  is a consequence relation or operation iff it satisfies the following conditions for all  $X \subseteq \mathcal{L}$ :*

- (T1)  $X \subseteq \mathcal{C}_n(X)$
- (T2) if  $X \subseteq Y$  then  $\mathcal{C}_n(X) \subseteq \mathcal{C}_n(Y)$
- (T3)  $\mathcal{C}_n(\mathcal{C}_n(X)) \subseteq \mathcal{C}_n(X)$
- (PF)  $\mathcal{C}_n(X) = \bigcup \{\mathcal{C}_n(Y) \mid Y \subseteq X; Y \text{ finite}\}$

REMARK 2.3.–

– Intuitively,  $\mathcal{C}_n$  maps a set of words from language  $\mathcal{L}$  to the set (i.e. the union) of their consequences.

– (T2) must be discarded to capture so-called non-monotonic logics. □

The notion of a consequence relation is the same as what is called *closure operation* in algebra.

DEFINITION 2.5.– (closure operation). *Given a set  $E$ , an operation  $\mathcal{C}: \mathcal{P}(E) \longrightarrow \mathcal{P}(E)$  is a closure operation iff for all  $X, Y \subseteq E$ :*

- $\mathcal{C}_1: X \subseteq \mathcal{C}(X)$
- $\mathcal{C}_2: \mathcal{C}^2(X) = \mathcal{C}(X)$
- $\mathcal{C}_3: \text{if } X \subseteq Y \text{ then } \mathcal{C}(X) \subseteq \mathcal{C}(Y)$

$X \subseteq E$  is closed iff  $\mathcal{C}(X) = X$ .

G. Gentzen (1909–1945) introduced a notion similar to that of Tarski with the following notation (see definition 3.11):

$$P_1, P_2, \dots, P_m \vdash C_1, C_2, \dots, C_n$$

with the following informal meaning: the *conjunction* of the  $P_i$ 's ( $1 \leq i \leq m$ ) admits the *disjunction* of the  $C_j$ 's ( $1 \leq j \leq n$ ) as a consequence.

More generally, we may write:

$\mathcal{A} \vdash \mathcal{B}$ , where  $\mathcal{A}, \mathcal{B}$  are sets of words on a formal language.

DEFINITION 2.6.– (consequence relation (Scott)). Given a formal language  $\mathcal{L}$ , a relation  $\vdash \subset \mathcal{P}(\mathcal{L}) \times \mathcal{P}(\mathcal{L})$  is a consequence relation in  $\mathcal{L}$ <sup>11</sup> iff it satisfies the following conditions:

- $A, B \in \mathcal{L}$ .
- $\mathcal{A}, \mathcal{B}, \mathcal{A}', \mathcal{B}', \mathcal{C} \subseteq \mathcal{L}$
- $\mathcal{A}, \mathcal{B}: \mathcal{A} \cup \mathcal{B}$
- $\mathcal{A}, \mathcal{B}: \mathcal{A} \cup \{B\}$
- (R)  $\mathcal{A} \vdash \mathcal{A}$  if  $\mathcal{A} \neq \emptyset$  (reflexivity)
- (M) if  $\mathcal{A} \vdash \mathcal{B}$  then  $\mathcal{A}, \mathcal{A}' \vdash \mathcal{B}, \mathcal{B}'$  (monotonicity)
- (T) if  $\mathcal{A} \vdash \mathcal{B}, \mathcal{C}$  and  $\mathcal{A}, \mathcal{B} \vdash \mathcal{C}$  then  $\mathcal{A} \vdash \mathcal{C}$  (transitivity)

Thanks to properties (R), (M) and (T),  $\vdash$  can be viewed as a generalization of a partial order (see definition 3.23).

It is fairly simple to show that a consequence relation à la Scott is a consequence relation à la Tarski, and conversely.

It is also fairly simple to prove that these relations coincide with the semantical notion of a logical consequence in definition 3.8.

We may now return to the question at the beginning of this digression and provide an answer.

What is an abstract logic?

Answer: a couple  $\mathcal{LA} = (\mathcal{L}, \vdash_{\mathcal{L}})$ ,

where

$\mathcal{L}$  is a formal language

$\vdash_{\mathcal{L}}$  is a consequence relation.

The calculi that we shall study (semantic tableaux, resolution, etc.) are different implementations of a consequence relation.

REMARK 2.4.– Two oddities are worth mentioning: the existence of a journal named *Informal logic*, which deals with the study of fallacies and argumentations, and the fact that an author introduced (in 1957) the concept of *infra-logic* (which is closely related to fuzzy logic).  $\square$

---

<sup>11</sup> Scott actually named the relation an *entailment relation*.

Logic allows us to analyze the relationships between objects, events, etc. without carrying out any experiment.

EXAMPLE 2.7.– (What is of interest is the *form*)

- 1) All humans have a spinal cord.
- 2) All living beings that have a spinal cord are capable of acquiring conditioned reflexes.
- 3) Therefore, all humans are capable of acquiring conditioned reflexes.

has the same form as:

- 1) All pigs have four legs.
- 2) All living beings with four legs have wings.
- 3) Therefore, all pigs have wings.

Both reasonings are correct, although in the second one, one of the premises and the conclusion are unrealistic.

(1) and (2) are the premises of the reasoning and (3) is the *conclusion*. □

EXAMPLE 2.8.– (... but one should be cautious)

- 1) I saw the portrait of Alexander Fleming.
- 2) Alexander Fleming discovered penicillin.
- 3) Therefore, I saw the portrait of the discoverer of penicillin.

has the same form as:

- 1) I saw someone's portrait.
- 2) Someone discovered the wheel.
- 3) Therefore, I saw the portrait of the discoverer of the wheel.

The first is a correct reasoning, whereas the second is not. □

EXAMPLE 2.9.– (*All* the premises must be specified<sup>12</sup>).

- 1) I did not pay my estate tax in time.
- 2) Therefore, I will have to pay a fine.

From a logical point of view, the conclusion is erroneous, as there is no premise stating that those who do not pay their estate tax in time have to pay a fine (for

---

<sup>12</sup> Some logics, especially some used in AI and called *default logics* do not have this requirement.

example, maybe if this is the first time I am late, I will only receive a warning from the tax collector, etc.).  $\square$

REMARK 2.5.– The usage of premises that are not explicit, which is due to bias or habits, and do not hold in the context of the discourse are a frequent (and unconscious) cause of fallacious argumentations.  $\square$

We now consider a fundamental distinction.

### 2.1.5.3. *Theory and meta-theory (language and meta-language)*

We shall prove a precise definition of a formal theory in section 3.3, but we have already stated the basic idea here: a formal (or formalized) theory is a set of sequences of symbols (unless stated otherwise, we shall only consider finite sequences of symbols), called *well-formed formulas* (wffs) and some simple operations that can be applied to these sequences.

It is important to make the distinction between:

- what belongs to the formal theory;
- what concerns the formal theory, considered as a closed deductive system.

In other words, it is important to distinguish between:

- the object language (logic): theory;
- the observer’s language (logic): meta-theory.

EXAMPLE 2.10.– In set theory

$$A \cup (A \cap B) = A \text{ and}$$

$$A \cap (A \cup B) = A$$

are theorems in the theory, whereas the

duality principle: “replacing every  $\cup$  (respectively  $\cap$ ) by a  $\cap$  (respectively  $\cup$ ) in a theorem of set theory yields another theorem in set theory” is a theorem in the meta-theory, or a *meta-theorem*.  $\square$

Confusing a theory and a meta-theory can have unfortunate consequences.

### 2.1.6. *A few thoughts about logic and computer science*

Before proceeding with the study of logic, using an approach that suggests the importance of logic by itself, and with the goal of showing that historically logic did

not always have proponents<sup>13</sup>, we mention two thoughts about logic and refute one of them (in our opinion, the other one is simply a witticism made by a genius).

At a time when paradoxes were becoming increasingly popular, the great mathematician Henri Poincaré said:

Logic is no longer sterile, it generates contradiction!<sup>14</sup>

(It suffices to mention Gödel's incompleteness theorem to fend off such attacks).

More recently, in a book written by two internationally renowned mathematicians (P. Halmos and S. Givant), we can read:

... the study of logic does not help anyone to think, neither in the sense that it provides a source of new thoughts nor in the sense that it enables one to avoid errors in combining old thoughts. Among the reasons for studying logic are these: (i) it is interesting in its own right, (ii) it clarifies and catalogues the methodology of mathematics, (iii) it has mathematical applications (for instance, in algebra, analysis, geometry, and set theory), and (iv) it is itself a rich application of some of the ideas of abstract algebra.

Of course, we agree with the four reasons given to study logic.

As far as the first sentence is concerned, we could also say (for example): *linguistics never taught anyone how to speak*.

From a more technical point of view, we can say that this vision is too restricted, in particular, it forgets that logic is a foundation for machines that help in thought operations (and therefore, that help to think). As a great French mathematician (A. Connes, recipient of the Fields medal) once said:

The verification process is extremely painful, because we are afraid we may have made a mistake. It is actually the most nerve-racking phase, because we cannot tell whether our intuition is correct... just as for dreams, intuition is easily mistaken. I remember having spent an entire month verifying a result: I would obsessively review every single detail of the proof, although this task could actually be entrusted to a computer which would verify the logic of the reasoning.

---

13 During the 4th Century, the Chinese thought that discursive reasoning was not a reliable way of grasping reality. In support of this statement, they cited the fallacious reasonings made by sophists, which led to conclusions that were obviously false.

14 Actually, he used the term "logicism" (which is one of the schools of thought in the philosophy of mathematics, to which is opposed, e.g. intuitionism).

A convincing example of the utility of logic (especially when it is handled by computer programs) is the proof of an old conjecture by Kepler (on sphere packing). The proof of this conjecture was produced in 1998, thanks to human and computer means, it was 250 pages long.

Those specialists who analyzed this proof concluded that it was correct... with a degree of certainty of 99%! A project (expected to last several years) meant to produce (using a computer) a formal proof of Kepler's conjecture was initiated in 2003.

Of course, logic does not claim (at least in principle) to capture all the operations of the mind (using images, discovery of regularities, causality, different kinds of analogies<sup>15</sup>, etc.).

## 2.2. Some historic landmarks

We cite some authors, among the most important, who contributed to the construction of Western logic (and thus of mathematical logic): Aristotle, Euclid, Leibniz, Boole, De Morgan, Frege, Whitehead, Russell, Tarski, and Gödel.

There is no written evidence of a theoretical presentation of logic before Aristotle. Before him (e.g. in Plato's dialogues), there were argumentations, but we are not concerned with the way to carry out a debate.

The works of Aristotle on logic reached us within a collection of texts called *Organon*, which means "instrument" (Aristotle considered logic as a preparatory discipline).

He introduced syllogisms.

This term, which has been used in everyday language for a long time, globally identifies every rigorous reasoning that does not use any implicit proposition.

He proposed categorical syllogisms, which are forms of reasoning that exclusively rely on categorical propositions (see section 2.2).

Aristotle considered four kinds of propositions (that were named *categorical*): universal affirmative propositions, universal negative propositions, particular affirmative propositions and particular negative propositions. The scholastics named them **A**, **E**, **I** and **O**, respectively (from Latin: *AffIrmo et nEgO*):

---

<sup>15</sup> It is worth mentioning that dogmatics (a school of medicine during the 2nd Century) used analogical inference, and that a theory of analogy was explored by scholastic logic (between the 12th and 15th Centuries).

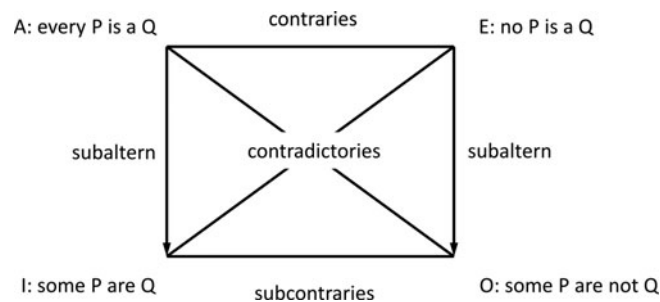
- A:** Every P is a Q
- E:** No P is a Q
- I:** Some P are Q
- O:** Some P are not Q

DEFINITION 2.7.– *Two propositions are:*

- *contradictories* iff they cannot be both true or both false;
- *contraries* iff they can both be false, but they cannot both be true;
- *subcontraries* iff they can both be true, but they cannot both be false;
- *subalterns*: a proposition Q is subaltern of a proposition P iff Q is necessarily true every time P is true and P is necessarily false every time Q is false.

We assume that there exist objects that satisfy property P.

For categorical propositions, this leads to the diagram known as the *square of opposition*, which establishes relations between any two categorical propositions:



The *theory of syllogisms (syllogistic theory)* is (generally considered as) Aristotle’s most important contribution to logic. Aristotle’s syllogistic theory was one of the pillars of logic for twenty centuries.

DEFINITION 2.8.– (syllogism) *A categorical syllogism is an argumentation containing only propositions A, E, I, and O, with two premises and one conclusion, and containing three terms that each appear once (and only once) in exactly two propositions*<sup>16</sup>.

<sup>16</sup> It is common to use *enthymemes*, which are abbreviated syllogisms with only one premise (the other one being implicit) and a conclusion. For example (when speaking about someone), “he is moving, therefore he is alive”.

EXAMPLE 2.11.—

*If all birds are animals  
and all sparrows are birds,  
then all sparrows are animals.*

*Animals* is the major term, *birds* is the middle term, and *sparrows* is the minor term.  $\square$

REMARK 2.6.— The main problem in syllogistic theory was to distinguish between correct syllogisms (reasonings) and incorrect ones.

Aristotle classified reasonings into 256 possible syllogisms, 24 of which were correct.  $\square$

The number 256 is simply obtained by enumerating all possibilities (Maj, Mid, Min, respectively, denote the major, middle and minor terms in a syllogism):

Premise 1	Premise 2
(Maj-Mid)	(Min-Mid)
(Mid-Maj)	(Min-Mid)
(Maj-Mid)	(Mid-Min)
(Mid-Maj)	(Mid-Min)

i.e. four possibilities.

For each of these possibilities, there are 16 possible combinations of the two premises: AA, AE, AI, AO, EA, EE, EI, EO, IA, IE, II, IO, OA, OE, OI, and OO.

There are also four possible conclusions (that correspond with the four categorical propositions). There are therefore  $4 \times 16 \times 4 = 256$  possible syllogisms.

REMARK 2.7.— During the 18th Century, the philosophies of Descartes, Locke, Hobbes, etc. led to the development of conceptions that were contrary to Aristotle's philosophy, and to a loss of interest in syllogisms.

Leibniz rehabilitated Aristotelian syllogisms.

Syllogistic theory is still used nowadays (with a mathematical formalism). We have the following translations (see Chapter 5 and remark 5.27):

(A) *Every P is a Q*:  $\forall x(P(x) \Rightarrow Q(x))$

(E) *No P is a Q*:  $\forall x(P(x) \Rightarrow \neg Q(x))$

(I) *Some P are Q*:  $\exists x(P(x) \wedge Q(x))$

(O) *Some P are not Q*:  $\exists x(P(x) \wedge \neg Q(x))$   $\square$



Below are some classical examples:

EXAMPLE 2.12.–

$$\frac{\begin{array}{l} \text{All } \textit{mammals} \text{ are animals} \\ \text{All } \textit{men} \text{ are } \textit{mammals} \end{array}}{\text{All } \textit{men} \text{ are animals}}$$

this corresponds to what is known as Barbara's syllogism:

$$\frac{\begin{array}{l} \text{Every B is a C} \\ \text{Every A is a B} \end{array}}{\text{Every A is a C}}$$

another syllogism:

$$\frac{\begin{array}{l} \text{Every A is a B} \\ \text{Some A are C} \end{array}}{\text{Some C are B}}$$

□

Some reasonings, although considered as correct, cannot be identified so using Aristotle's syllogistic. For example:

$$\frac{\begin{array}{l} A \vee B \\ \neg A \end{array}}{B}$$

or:

$$\frac{\text{Every p is Q or R}}{\text{Every p that is not Q is R}}$$

A clue of the influence of Aristotelian syllogisms through time is the fact that in the 19th Century, machines that could automate these syllogisms were constructed.

Along with the notion of a syllogism, Aristotle<sup>17</sup> defined the notion of a variable (see digression 3.3), and he formulated two laws:

– contradiction law: two contradictory statements cannot both be (simultaneously) true. This contradiction principle is the fundamental principle of thought according to Aristotle. He considers it the most unquestionable principle. It is the principle of being itself (*Nobody could ever think that one same thing could be and not be*), the most certain principle, and it is of course impossible to prove;

---

<sup>17</sup> Some authors credit the law of excluded middle to Zeno of Elea.

– law of excluded middle: if two statements are contradictory, then one of them must be true (there cannot be a third possibility)<sup>18</sup>.

*The contradiction principle does not answer the question whether it is possible to know if a middle term between the assertion and the negation is possible. The law of excluded middle rejects this possibility*<sup>19</sup>.

One may wonder what is the relationship between these two principles. According to Aristotle, the negation of the contradiction principle entails the negation of the law of excluded middle. But this does not mean that the law of excluded middle can be deduced from the contradiction principle. *These are independent principles.*

In logic, as in other domains in science, seemingly obvious principles become much less so when they are analyzed in detail. Consider the following proposition:

The Greek Aristarchus of Samos hypothesized in 250 BC that the Earth revolved around the Sun.

Intrinsically, (from an ontological point of view), this proposition admits either the value **T** or **F**, and can take no other value.

But what if we consider it from the point of view of what we know (epistemic point of view)? Most people would probably say they do not know whether it is **T** or **F**, and would rather assign it a third value: indeterminate (often denoted by  $\perp$ ). The method used to try to assign a truth value to such a proposition is fundamentally different from the one used to assign a truth value to propositions such as:

$2^{30402457} - 1$  is a prime number.

or

7427466391 is the first prime number made of ten consecutive digits from the decimals of  $e$ .

Around 453–222 BC, a sect of Chinese preacher–monks, called Mohists (disciples of Mozi) came up with parts of a logic that was quickly forgotten.

As we have already mentioned, logic was influenced by Aristotle for a very long time. In the 17th Century, Kant stated that logic had not made a single step forward,

---

<sup>18</sup> We have already seen that for constructivists, this law cannot be applied to just any assertion.

<sup>19</sup> We shall return to these two laws when we shall present logics other than classical logic (multi-valued logic, fuzzy logic).

and that it was finished and perfect. He also attacked formalization attempts, as he believed it was impossible to replace speculation by symbol manipulation.

Something similar took place for Euclidean geometry. Euclid was the first to propose what we now call, after a few modifications, an *axiomatization*. It was an axiomatization for elementary geometry. Until the discovery of non-Euclidean geometries (19th Century), this axiomatization was considered to be a completed model of the real world, and the only axiomatization to be coherent (or consistent).

Leibniz (17th to 18th Century) believed that logic could be transformed into a calculus (with all the advantages of working with symbols) that could be applied to all human activities.

Leibniz worked before Frege on the notion of a formal system.

Leibniz's project was (at least partially) fulfilled by G. Boole (19th Century). One of his publications was a cornerstone of the separation of logic and philosophy. Boole discovered fallacies in philosophy books by using algebra.

De Morgan (19th Century) showed that it was impossible for all argumentations to be reduced to argumentations that exclusively relied on categorical statements.

Frege (19th to 20th Century) was inspired by Leibniz's ideas, as the latter had developed a language for "pure thought". He introduced formal systems (in their current form) and the theory of quantification. The symbolism he used was too cumbersome and was discarded later (we saw an example of his formalism in section 2.1).

Whitehead and Russell (19th to 20th Century) are the authors of the monumental *Principia Mathematica* and Russell is the author of the famous paradox that bears his name and that evidenced how dangerous it could be to allow a set to be defined by just any property (existence of contradictions in naive set theory, see section 2.1.2).



## Chapter 3

# Propositional Logic

The first logic we shall study is propositional logic (PL or PC), and although this logic has a limited expressive power, its wide variety of applications along with its role in the foundations of automation of first-order logic makes it an essential topic to study.

DEFINITION 3.1.– (proposition 1). *A statement or an expression (i.e. a syntactically correct sequence of characters) that expresses a relation between several objects (terms) is a proposition.*

For most of the material in this course, the following definition will be sufficient.

DEFINITION 3.2.– (proposition 2). *A proposition is a statement or expression to which one (and only one) true value can be assigned.*

REMARK 3.1.– Later on (see section 10.1), we shall see other possible values that can be assigned to a proposition.  $\square$

EXAMPLE 3.1.– (declarative and imperative sentences).

“The factorial of 0 is 1” is a proposition.

“The factorial of 1 is 1” is a proposition.

“The factorial of  $n(n > 1)$  is  $n - 1$  times the factorial of  $n - 2$ ” is a proposition.

“Go to label LAB” is *not* a proposition.

“Store value 3 at the index labelled by  $x$ ” is not a proposition.  $\square$

We shall use two approaches to study PL (one semantical and the other syntactic). In both cases, we need a language.

### 3.1. Syntax and semantics

The key notion in the semantical approach is that of interpretation. In the syntactical approach (see section 3.3), it will be that of symbolic manipulation (which implies: independent of all meaning).

The language is the one defined by the syntax of PC (PL).

DEFINITION 3.3.— *The only accepted formulas are the ones with a syntax compliant with the rules given below. They are called well-formed formulas, or simply wffs.*

*Let  $\Pi$  denote a denumerably infinite set of basic formulas (or atomic formulas, or elementary formulas, or propositional symbols) that shall be noted, for example,  $P_1, P_2, \dots$ , or  $P_1, P_2, \dots$ , or  $P, Q, R, \dots$ .*

*The set of propositional formulas (denoted by  $\mathcal{L}_0$ ) is the smallest set containing the elements in  $\Pi$ , which is closed for the rule:*

*If  $\mathcal{A}$  and  $\mathcal{B}$  are propositional formulas, then:  
 $\neg \mathcal{A}$ ,  $\mathcal{A} \wedge \mathcal{B}$ ,  $\mathcal{A} \vee \mathcal{B}$ ,  $\mathcal{A} \Rightarrow \mathcal{B}$ , and  $\mathcal{A} \Leftrightarrow \mathcal{B}$  are also propositional formulas.*

*$\neg$ ,  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ , and  $\Leftrightarrow$  are called logical connectives or simply connectives.*

The *hierarchy* (or *priority*) of connectives is, in decreasing order (which means that negation is the first to be applied):

- $\neg$  negation
- $\wedge$  conjunction
- $\vee$  disjunction
- $\Rightarrow$  implication
- $\Leftrightarrow$  equivalence

For one same operator: from left to right.

For example:

$$A \vee \neg B \Leftrightarrow C \text{ is the same as } (A \vee (\neg B)) \Leftrightarrow (C)$$

REMARK 3.2.— (the set of wffs of PL is *denumerably infinite*). The set of wffs of PL is *denumerably infinite*: there are many basic symbols denumerably infinite and it is possible to come up with a procedure that enumerates wffs.

Every infinite set of wffs of PL will be a subset of a denumerably infinite set, and will therefore also be denumerably infinite.  $\square$

DEFINITION 3.4.– (semantics of PL (of PC)). An  $n$ -ary truth function is a function:

$$\{\mathbf{T}, \mathbf{F}\}^n \longrightarrow \{\mathbf{T}, \mathbf{F}\} \quad n \geq 1$$

Among the  $2^{(2^2)}$  possible binary truth functions, the following set of connectives is usually used:  $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ . It is sufficient to express all possible binary functions (see exercise 3.2).

DEFINITION 3.5.– Let  $\Pi$  denote the set of basic formulas in the language. An interpretation  $I$  is a function:

$$I: \Pi \longrightarrow \{\mathbf{T}, \mathbf{F}\}$$

Practically speaking, we shall generally be interested in finite sets of formulas, and it will suffice to consider  $\Pi$  restricted to the basic formulas occurring in the formulas under consideration (see definition 3.8).

EXERCISE 3.1.– How many interpretations are there for PL:

- 1) A finite number?
- 2) A denumerably infinite number?
- 3) An uncountably infinite number? □

DEFINITION 3.6.– (interpretation of non-elementary wffs: truth tables). We define  $\bar{I}$ , the extension of  $I$  to the set of propositional formulas, as follows:

- 1)  $\bar{I} = I(A)$  if  $A$  is an atomic formula
- 2)  $\bar{I}(\neg A) = \begin{cases} \mathbf{F} & \text{if } \bar{I}(A) = \mathbf{T} \\ \mathbf{T} & \text{if } \bar{I}(A) = \mathbf{F} \end{cases}$
- 3)  $\bar{I}(A \wedge B) = \begin{cases} \mathbf{T} & \text{if } \bar{I}(A) = \bar{I}(B) = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases}$
- 4)  $\bar{I}(A \vee B) = \begin{cases} \mathbf{T} & \text{if } \bar{I}(A) = \mathbf{T} \text{ or } \bar{I}(B) = \mathbf{T} \text{ or both} \\ \mathbf{F} & \text{otherwise} \end{cases}$
- 5)  $\bar{I}(A \Rightarrow B) = \begin{cases} \mathbf{T} & \text{if } \bar{I}(A) = \mathbf{F} \text{ or } \bar{I}(B) = \mathbf{T} \\ \mathbf{F} & \text{otherwise} \end{cases}$
- 6)  $\bar{I}(A \Leftrightarrow B) = \begin{cases} \mathbf{T} & \text{if } \bar{I}(A) = \bar{I}(B) \\ \mathbf{F} & \text{otherwise} \end{cases}$

$\bar{I}(A)$ , which shall also be noted as  $\mathcal{E}(A, I)$ , is similar to the well-known truth tables

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

REMARK 3.3.– Compare these with the inference rules of semantic tableaux (see section 3.2). □

REMARK 3.4.– (history of truth tables). The introduction of truth tables in modern logic is attributed to C.S. Peirce, who borrowed them from Stoicians (4th Century BC), and they have also been attributed to L. Wittgenstein.

We know that they were also known by the Megarians (school of philosophy founded by Euclid of Megara, 5th to 4th Century BC), and, in particular, we know that Philo of Megara proposed truth functions. This school, to which Eubulides belonged (see example 8.1), was interested in “everyday” argument and in paradoxes.

Truth tables were a means to decide the truth value of any wff. □

The translation of sentences from everyday language to the language of PC comes with its share of problems, and we must be cautious and take into account, for example, the habits of a given language.

EXAMPLE 3.2.–

- a) If it rains, then I will go to the movies.
- b) This afternoon, at 2 pm, I will go to the movies or to play soccer.

If we wished to translate the (most probable) semantics of (a), we would use a  $\Leftrightarrow$  (but not a  $\Rightarrow$ ) because according to *habits*, what we mean is “if it rains, then I will go to the movies, and if it does not rain, then I will not go to the movies”.

(b) Should of course be translated using and exclusive or.

If  $A$  then  $B$  can be translated by  $(A \Rightarrow B) \wedge (\neg A \Rightarrow \neg B)$ .

When we say “some  $P$  are  $Q$ ” we also mean “some  $P$  are not  $Q$ ”.

In logic (as in mathematics), we assume that premises are interpreted in a *minimal* way, that is, if  $P$  then  $Q$  is translated by  $P \Rightarrow Q$ ; there exists  $x$  such that, we assume there exists such an  $x$ , etc. □



Just as in the case of a function, we can define the extension and the restriction of an interpretation.

### 3.1.1. Language and meta-language

$\models \mathcal{A}$  :  $\mathcal{A}$  is valid, that is, every interpretation of  $\mathcal{A}$  is a model of  $\mathcal{A}$  (one such example is  $\mathcal{A}$ :  $A \vee \neg A$ ).

$\mathcal{A} \models \mathcal{B}$ : every model of  $\mathcal{A}$  is also a model of  $\mathcal{B}$

(one such example is  $\mathcal{A}$ :  $A$  and  $\mathcal{B}$ :  $A \vee \alpha$  with  $\alpha : \bigvee_{i=1}^n B_i$ ).

REMARK 3.5.– Note the difference between  $\Rightarrow$  and  $\models$ .  $\Rightarrow$  is a symbol of the language, whereas  $\models$  is a symbol of the meta-language.  $\square$

The notation  $\models_{\mathcal{I}} F$  means “interpretation  $\mathcal{I}$  is a model of  $F$ ”.

The notation  $F \models_{\mathcal{I}} G$ , or equivalently (see exercise 3.14)  $\models_{\mathcal{I}} F \Rightarrow G$ , means “interpretation  $\mathcal{I}$  is a model of  $F$  and  $G$ ”

Although it is trivial, the following meta-theorem is very important.

META-THEOREM 3.1.–  $H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$  iff

$H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  is unsatisfiable (or contradictory, i.e. impossible to evaluate to  $\mathbf{T}$ ).

Since the origin of logic, the interest of considering sets of truth values other than  $\{\mathbf{T}, \mathbf{F}\}$  has been widely acknowledged.

Among others, linguists propose examples of propositions such as:

*The current king of France is bald.*

*The current king of France is not bald.*

*Michael (who has always been a vegan) stopped eating meat.*

*Michael (who has always been a vegan) did not stop eating meat.*

that are neither  $\mathbf{T}$  nor  $\mathbf{F}$ .

Although we shall not mention logics with more than two truth values before Chapter 10, we show an example of a truth table for a logic with three truth values, the value  $\perp$  denoting an indeterminate value.

EXAMPLE 3.3.– (truth tables with partial information (Kleene)).

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>V</b>
<b>T</b>	<b>F</b>		<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	$\perp$		$\perp$	<b>T</b>	$\perp$	$\perp$
<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>F</b>		<b>F</b>	<b>F</b>	<b>T</b>	<b>V</b>
<b>F</b>	$\perp$		<b>F</b>	$\perp$	<b>T</b>	$\perp$
$\perp$	<b>T</b>	$\perp$	$\perp$	<b>T</b>	<b>T</b>	$\perp$
$\perp$	<b>F</b>		<b>F</b>	$\perp$	$\perp$	$\perp$
$\perp$	$\perp$		$\perp$	$\perp$	$\perp$	$\perp$

□

DEFINITION 3.7.– A set of connective is adequate iff it permits to express all truth functions.

EXERCISE 3.2.– Prove that the sets of connectives below are adequate. More precisely, design an algorithm that takes an arbitrary truth table as an input (graph of a truth function), and as an output produces the same function, defined using a wff constructed with (exactly) the same propositional variables and (exclusively) the given set of connectives.

- a)  $\{\neg, \wedge, \vee\}$
- b)  $\{\neg, \wedge\}$
- c)  $\{\neg, \vee\}$
- d)  $\{\neg, \Rightarrow\}$
- e)  $\{\mid\}$  % see the definition below
- f)  $\{\downarrow\}$  % see the definition below

$P$	$Q$	$P \mid Q$	$P \downarrow Q$
<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>

□

EXERCISE 3.3.– A wff of PL is *positive* if it contains only propositional symbols and the connectives  $\wedge$  and  $\vee$ .

Consider an  $n$ -ary truth function  $f: \{\mathbf{T}, \mathbf{F}\}^n \rightarrow \{\mathbf{T}, \mathbf{F}\}$

such that:

$f(\mathbf{F}, \mathbf{F}, \dots, \mathbf{F}) = \mathbf{F}$  %, that is if all its arguments are **F** then its value is **F**;

$f(\mathbf{T}, \mathbf{T}, \dots, \mathbf{T}) = \mathbf{T}$  %, that is if all its arguments are  $\mathbf{T}$  then its value is  $\mathbf{T}$ .

$f$  is the truth table of a positive wff of PL.

Do you believe:

a) the assertion above is true?

b) the assertion above is false?

EXERCISE 3.4.– Give the truth tables of the following wffs:

a)  $P \wedge (P \Rightarrow Q) \Rightarrow Q$

b)  $(P \Rightarrow Q) \wedge (\neg Q \Rightarrow \neg P)$

c)  $\neg A \wedge (A \vee B) \Rightarrow B$

d)  $A \Rightarrow A \vee B$

e)  $(P \Rightarrow Q) \wedge \neg Q \Rightarrow \neg P$

EXERCISE 3.5.– Answer questions (a) and (b) given below without entirely constructing the truth tables and without applying any transformation to the formula. This exercise is an introduction to a method that will be studied in section 3.2.

a) Is the following wff always  $\mathbf{T}$ ? Always  $\mathbf{F}$ ?

$$(((P \Rightarrow Q) \wedge (P \Rightarrow \neg R)) \wedge (R \Rightarrow P)) \wedge (\neg P \Rightarrow Q) \Rightarrow (S \Rightarrow (R \vee T))$$

b) Can the following wff be evaluated to  $\mathbf{F}$ ?

$$(((A \Rightarrow B) \Rightarrow (\neg C \Rightarrow \neg D)) \Rightarrow C) \Rightarrow E \Rightarrow ((E \Rightarrow A) \Rightarrow (D \Rightarrow A)) \quad \square$$

The following definitions (that were regrouped for the sake of simplicity) introduce some fundamental notions:

DEFINITION 3.8.– (important concepts and terminology). (*The definitions of interpretations and evaluations have already been presented, but they are repeated here so that they can be regrouped with other definitions.*)

– A wff  $F$  is in negative normal form (nnf) iff the negation symbols that occur in  $F$  are exclusively applied to atomic formulas.

– A wff is in conjunctive normal form (cnf) or in clausal form iff it is of the form:

$$C_1 \wedge C_2 \wedge \dots \wedge C_n \quad (1 \leq i \leq n), \quad \text{we note } \bigwedge_{i=1}^n C_i$$

where each  $C_i$  is called a conjunct (or a clause)<sup>1</sup> and is of the form:

$$A_1 \vee A_2 \vee \dots \vee A_{m_i}, \quad \text{we note } \bigvee_{j=1}^{m_i} A_j$$

where  $A_i \neq A_j$  for  $i \neq j$  (clauses are also defined as sets of literals).

The length of a clause is the number of literals it contains.

A clause of length 1 (i.e.  $m_i = 1$ ) is called a unit clause.

The cnf is associated with the set of clauses  $\{C_1, C_2, \dots, C_n\}$ .

– The wff is in disjunctive normal form (dnf) iff it is of the shape:

$$D_1 \vee D_2 \vee \dots \vee D_n \quad (1 \leq i \leq n), \quad \text{we note } \bigvee_{i=1}^n D_i$$

where each  $D_i$  is called a disjunct and is of the form:

$$A_1 \wedge A_2 \wedge \dots \wedge A_{m_i} \quad \text{disjunctive normal form} \quad \text{we note } \bigwedge_{j=1}^{m_i} A_j.$$

For homogeneity reasons, we shall allow conjunctions (disjunctions) of empty sets. In this case, the conjunction (disjunction) will not add any symbol to the syntax of the wff it occurs in.

– In both cases,  $A_j$  is a propositional atom (i.e. a basic propositional symbol) or the negation of a propositional atom, and will be called a literal.

– If  $L$  (respectively,  $\neg L$ ) is a literal, then  $\neg L$  (respectively,  $L$ ) is called the complementary literal of  $L$  (respectively,  $\neg L$ ). It is denoted by  $L^c$ .

$L$  is a positive literal and  $\neg L$  a negative literal.

We will also talk of the sign of a literal (respectively, positive and negative).

– If all literals in a clause are positive (respectively, negative), then the clause is positive (respectively, negative).

– A wff  $G$  is the cnf (respectively, dnf) of a wff  $F$  iff  $G$  and  $F$  are equivalent (i.e.  $F \Leftrightarrow G$  is a tautology, see below), and  $G$  is in cnf (respectively, dnf).

– An interpretation  $I$  of a language (a wff  $\mathcal{A}$ ) is an application:

$$I : \Pi \longrightarrow \{\mathbf{T}, \mathbf{F}\}$$

where  $\Pi$  is the set of propositional symbols (basic formulas, atomic formulas, and elementary formulas) of the language (of  $\mathcal{A}$ ). If the domain of the interpretation is restricted to a proper subset of  $\Pi$ , then  $I$  is a partial interpretation

In particular,

---

<sup>1</sup> The term clause is also used in the simplification of truth functions and test of their validity, to designate the disjuncts of a dnf, that is, the conjunction of literals (see remark 3.36).

– The interpretation of a formula  $\mathcal{F}$  (respectively, of a set of formulas  $\mathcal{S}$ ) is the restriction of the language that  $\mathcal{F}$  (respectively,  $\mathcal{S}$ ) belongs to the set of propositional symbols of  $\mathcal{F}$  (respectively,  $\mathcal{S}$ ). This set will generally be denoted by  $\text{Propset}(\mathcal{F})$  (respectively,  $\text{Propset}(\mathcal{S})$ ).

– A partial interpretation of a formula  $\mathcal{F}$  is the restriction of the interpretation of the language that  $\mathcal{F}$  belongs to a set  $\mathcal{D}$  in which  $\mathcal{D} \subsetneq \text{Propset}(\mathcal{F})$ .

– An evaluation  $\mathcal{E}$  of a wff  $\mathcal{A}$  for an interpretation  $I$  is the truth value of  $\mathcal{A}$  for  $I$ . It is denoted by  $\mathcal{E}(\mathcal{A}, I)$  and sometimes by  $\bar{I}(\mathcal{A})$ .

– A model of a wff  $\mathcal{A}$  (a set of wffs  $\mathcal{S}$ ) is an interpretation  $I$  of  $\mathcal{A}$  (of  $\mathcal{S}$ ) such that  $\mathcal{E}(\mathcal{A}, I) = \mathbf{T}$  ( $\mathcal{E}(\mathcal{A}, I) = \mathbf{T}$  for all  $\mathcal{A} \in \mathcal{S}$ ). We say that  $I$  satisfies  $\mathcal{A}$  ( $\mathcal{S}$ ).

– A counter model (counter example) of a wff  $\mathcal{A}$  is an interpretation  $I$  of  $\mathcal{A}$  (of  $\mathcal{S}$ ) such that  $\mathcal{E}(\mathcal{A}, I) = \mathbf{F}$  ( $\mathcal{E}(\mathcal{A}, I) = \mathbf{F}$  for at least one  $\mathcal{A} \in \mathcal{S}$ ). We say that  $I$  falsifies  $\mathcal{A}$  ( $\mathcal{S}$ ).

– A wff is satisfiable or consistent or coherent iff it admits (at least) one model.

– A wff  $\mathcal{A}$  is unsatisfiable or contradictory iff it admits no models, that is, for all  $I$ :  $\mathcal{E}(\mathcal{A}, I) = \mathbf{F}$ .

– A wff  $\mathcal{A}$  is valid or tautological iff for all  $I$ :  $\mathcal{E}(\mathcal{A}, I) = \mathbf{T}$ .

We note  $\models \mathcal{A}$ .

– A wff  $\mathcal{B}$  is a logical consequence of a wff  $\mathcal{A}$ , denoted by  $\mathcal{A} \models \mathcal{B}$ , iff every model of  $\mathcal{A}$  is a model of  $\mathcal{B}$ .

These definitions naturally extend to sets of wffs.

We can therefore write  $\{P_1, P_2, \dots, P_n\} \models C$  or, as is more common,

$$P_1, P_2, \dots, P_n \models C.$$

EXERCISE 3.6.– For  $F, G$  in  $\mathcal{L}_0$ , we define the binary relation  $\preceq$ :  $F \preceq G$  iff  $F \models G$ .

Is  $\preceq$  an order relation? (Also see definition 3.26) □

How can interpretations be specified? The following example shows three common ways to do so.

EXAMPLE 3.4.– Consider the following formula:

$$\mathcal{A} : [((A \wedge B) \vee C) \Rightarrow (C \wedge \neg D)] \Leftrightarrow E$$

(1), (2), and (3) given below are three different notations that specify the same interpretation<sup>2</sup>,

<sup>2</sup> Strictly speaking, we should make the value of  $I$  on  $\Pi \setminus \{A, B, C, D, E, F\}$  explicit or clarify that it is an interpretation of the formula  $\mathcal{A}$ .

$$1) I = \{ \langle A, \mathbf{F} \rangle, \langle B, \mathbf{F} \rangle, \langle C, \mathbf{T} \rangle, \langle D, \mathbf{F} \rangle, \langle E, \mathbf{T} \rangle \}$$

$$2) I = \{ \neg A, \neg B, C, \neg D, E \}$$

$$3) I = \{ C, E \} \text{ (here, only those propositions that evaluate to } \mathbf{T} \text{ are mentioned)}$$

$$\bar{I}(\mathcal{A}) = \mathcal{E}(\mathcal{A}, I) = \mathbf{T}$$

For notations (1) and (2):

1) If  $\langle P, \mathbf{T} \rangle$  (respectively,  $\langle P, \mathbf{F} \rangle \in I$  then  $\langle P, \mathbf{F} \rangle$  (respectively,  $\langle P, \mathbf{T} \rangle \notin I$

2) If  $P$  (respectively,  $\neg P \in I$  then  $\neg P$  (respectively,  $P \notin I$

( $I$  is an *application*). Of course, this remark is not necessary for notation 3.  $\square$

EXERCISE 3.7.– Verify that the following formulas (paradoxes of material implication) are tautologies:

$$a) P \Rightarrow (Q \Rightarrow P)$$

$$b) \neg P \Rightarrow (P \Rightarrow Q)$$

Do they have a translation in everyday language?  $\square$

REMARK 3.6.– The following tautologies are often used in mathematics, in the statement of theorems:

$$A \Rightarrow (B \Rightarrow C) \Leftrightarrow (A \wedge B \Rightarrow C)$$

$$A \Rightarrow (B \Rightarrow (C \Rightarrow D)) \Leftrightarrow (A \wedge B \wedge C \Rightarrow D) \quad \square$$

EXERCISE 3.8.–

– Is the following reasoning (or argument) correct?

– How can a correct reasoning be characterized? In other words, what relationship must hold between the premises and the conclusion for the reasoning to be correct?

– Using your definition of a correct reasoning, can you verify that the syllogisms of example 2.12 are correct reasonings?

If life has a meaning, but necessarily ends by death, then life is sad.

If life goes on after death, but does not have a meaning, then life is a cosmic joke.

If life had a meaning and went on after death, then angst would not exist.

If life is not a cosmic joke, then it is not sad.

Angst exists.

If life is sad or if it is a cosmic joke, then it is not beautiful.

Therefore life is ugly.  $\square$

REMARK 3.7.– Perhaps here is the best place to recall the following quote from Wittgenstein: “Logic is a vast tautology that does not state anything about the world, it simply expresses the equivalence between propositions”.  $\square$

A reasoning is a set of premises and a conclusion. Reasonings are usually represented by a column of premises that are separated from the conclusion by a horizontal line.

EXERCISE 3.9.– Are the following reasonings correct?

a)

$$\begin{array}{l} A \Rightarrow B \\ A \Rightarrow C \\ \hline \neg(B \vee C) \\ D \end{array}$$

b)

$$\begin{array}{l} A \Rightarrow B \\ B \Rightarrow C \\ C \Rightarrow D \\ \neg D \\ \hline A \vee E \\ E \end{array}$$

The notion of the *normal form* of a formula is very important for at least two reasons: the first one is that it permits us an economy of thought, as we can always assume (once we have proved the existence of this normal form) that the formula under consideration is already under normal form. The second reason is that when this normal form is unique, we can prove the equivalence (equality) of syntactically different formulas by a reduction to this normal form.

In definition 3.8, we introduced two normal forms for formulas in propositional logic: the cnf and the dnf. The following rules allow us to obtain these normal forms.

### 3.1.2. Transformation rules for cnf and dnf

– Step 1: elimination of  $\Leftrightarrow$  and  $\Rightarrow$

$$1) A \Leftrightarrow B \longrightarrow (A \Rightarrow B) \wedge (B \Rightarrow A)$$

$$2) (A \Rightarrow B) \longrightarrow \neg A \vee B$$

– Step 2: put  $\neg$  next to the atoms

- 1)  $\neg(\neg A) \longrightarrow A$
- 2)  $\neg(A \vee B) \longrightarrow \neg A \wedge \neg B$
- 3)  $\neg(A \wedge B) \longrightarrow \neg A \vee \neg B$

– Step 3: distributivity of  $\vee$  and  $\wedge$

- 1)  $A \vee (B \wedge C) \longrightarrow (A \vee B) \wedge (A \vee C)$
- 2)  $A \wedge (B \vee C) \longrightarrow (A \wedge B) \vee (A \wedge C)$

The following rules are often used:

- $$\neg(A \Rightarrow B) \longrightarrow A \wedge \neg B$$
- $$A \vee A \longrightarrow A$$
- $$A \wedge A \longrightarrow A$$
- $$(A \vee \neg A) \wedge B \longrightarrow B$$
- $$(A \wedge \neg A) \vee B \longrightarrow B$$
- $$A \vee B \longrightarrow B \vee A$$
- $$A \wedge B \longrightarrow B \wedge A$$

REMARK 3.8.– (3-cnf). We can prove (by adding propositional variables that are used to introduce definitions) that every cnf formula  $F$  can be transformed into another cnf  $F_{\leq 3}$  with at most three literals in every clause, such that  $F_{\leq 3}$  is satisfiable if and only if  $F$  is satisfiable.  $\square$

EXERCISE 3.10.–

- a) Construct the cnf of  $(P \wedge (Q \Rightarrow R) \Rightarrow S)$
- b) Construct the dnf of  $(P \vee \neg Q) \Rightarrow R$
- c) If  $\mathcal{CNF}$  denotes the set of cnf formulas and  $\mathcal{DNF}$  denotes the set of dnf formulas, do we have  $\mathcal{DNF} \cap \mathcal{DNF} = \emptyset$ ?
- d) Given a cnf, are all the transformation rules required to obtain a dnf? Construct the dnf of:

$$(A \vee \neg B \vee C) \wedge (\neg D \vee E) \wedge (F \vee \neg G \vee H)$$

- e) Is the cnf (respectively, dnf) of a wff unique?  $\square$

REMARK 3.9.– Sometimes the transformation rules generate redundancies that can potentially (and artificially) increase the search space of a method using a normal form, as shown in the following example:

$$\neg(P \Leftrightarrow Q) \longrightarrow \neg[(P \Rightarrow Q) \wedge (Q \Rightarrow P)] \longrightarrow \neg[(\neg P \vee Q) \wedge (\neg Q \vee P)] \longrightarrow [(P \wedge \neg Q) \vee (Q \wedge \neg P)] \longrightarrow [(P \vee Q) \wedge \underline{(P \vee \neg P)} \wedge \underline{(\neg Q \vee Q)} \wedge (\neg Q \vee \neg P)]$$

The conjuncts that are underlined are tautologies that could be eliminated from the start by introducing a new rule:



$$\neg(P \Leftrightarrow Q) \longrightarrow (P \vee Q) \wedge (\neg P \vee \neg Q) \quad \square$$

Note that in the analysis of deductive argument performed below, two different strategies (or any combination of the strategies) can be considered. The word *strategy* means ordering and decrease (if possible) of the number of choices in non-deterministic problems (see section 3.8.1):

– forward chaining (*bottom up*): starting from the premises and by application of the inference rules (or by using the meaning of connectives), try to reach the conclusion. We have not yet defined what inference rules are (see definition 3.9), but for the time being it is sufficient to consider them as elementary reasoning steps (which corresponds to the formal definition). The important thing is to *declare* the elementary reasoning steps that is allowed to use.

*We move from the premises to the conclusion.*

– backward chaining (*top down*): starting from the conclusion (denoted by C) reason by saying: if we want to have C, it suffices to have A, if we want to have A, it suffices to have B, etc., and if possible, move to the premises.

*We move from the conclusion to the premises.*

EXERCISE 3.11.– Is the following reasoning correct?

If there is a unique norm to judge greatness in art, then both M and G cannot be great artists. If P or D are considered as great artists, then W is certainly not one. However, if W is not a great artist, then K or S are not great artists either. After all, G is not a great artist, but D and K are.

Therefore, there is no unique norm to judge greatness in art. □

EXERCISE 3.12.– Imagine a reasoning that is presented as usual by a set of premises and a conclusion. The premises and the conclusion are wffs of propositional logic.

$Propset(set - wff)$  is the set of propositional symbols in the set of wffs  $set - wff$ .

a) Assume that  $Propset(premises) \cap Propset(conclusion) = \emptyset$ . Can the reasoning be correct?

If this is not the case in general, are there particular cases in which this property holds?

b) Now imagine that the premises and the conclusion are in clausal form (see definition 3.8). This assumption does not incur any loss of generality (why?).

Given a set of clauses, a literal (see definition 3.8) whose negation does not occur in the other clauses is called a *pure literal*.

If a premise contains a pure literal, can the analysis of the reasoning be simplified?  
How? □

EXERCISE 3.13.– A crime occurred in a luxurious home and the police inspector in charge of the investigation reasons as follows:

If on the day of the crime, someone had asked the house servant the question “Why were you not at dinner the day before yesterday?”, she would have answered.

If she had answered, someone would have heard her.

The house servant was not heard.

If the house servant was neither seen nor heard, that is because she was polishing cutlery, and if she was polishing cutlery, then she was here on the day of the crime.

I, therefore, conclude that the house servant was in the house when the crime occurred.

a) What premise(s) should the police inspector have added for the reasoning to be correct?

b) Is there a unique possibility?

c) Can a correct reasoning be made incorrect by adding premises? □

The proof of the following meta-theorem is trivial.

META-THEOREM 3.2.– *A is valid iff  $\neg A$  is unsatisfiable (contradictory).*

EXERCISE 3.14.– Prove the following meta-theorem (deduction theorem – semantical version):

$\mathcal{A} \models \mathcal{B}$  iff  $\models \mathcal{A} \Rightarrow \mathcal{B}$  □

We will be interested in expressions of the form:

$H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$

where the  $H_i$ 's ( $1 \leq i \leq n$ ) and  $C$  are wffs.

These kinds of expressions correspond to the informal notion of a correct reasoning.

EXERCISE 3.15.– Prove the following meta-theorems:

a)  $H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$  iff  $\models H_1 \wedge H_2 \wedge \dots \wedge H_n \Rightarrow C$

Note that in mathematics, we usually write:

$$H_1 \wedge H_2 \wedge \dots \wedge H_n \Rightarrow C$$

with the implicit meaning:

$$\models H_1 \wedge H_2 \wedge \dots \wedge H_n \Rightarrow C.$$

b)  $H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$  iff  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  is unsatisfiable.

We will sometimes use the set-theoretical version of this statement:  $C$  is a logical consequence of  $\{H_1, H_2, \dots, H_n\}$  iff  $\{H_1, H_2, \dots, H_n, \neg C\}$  is unsatisfiable.

(This last meta-theorem is the well-known *reductio ad absurdum* proof technique: reaching a contradiction by negating the conclusion.)  $\square$

We have seen (see exercise 3.12) that what is of interest in a non-trivial reasoning are the propositional symbols that occur on the left-hand side and the right-hand side of  $\models$ : their interpretation on the left-hand side (respectively, right-hand side) fixes their interpretation on the right-hand side (respectively, left-hand side).

EXERCISE 3.16.– By assuming that:

$\mathcal{A} \not\models \mathcal{C}$  %, that is, not all models of  $\mathcal{A}$  are models of  $\mathcal{C}$ ,

and

$$\mathcal{A} \wedge \mathcal{B} \models \mathcal{C}.$$

Which of the following assertions is correct?

a)  $\mathcal{A} \models \mathcal{B}$ ;

b)  $\mathcal{A} \not\models \mathcal{B}$ .  $\square$

EXERCISE 3.17.– Let  $S$  denote a finite set of wffs  $S = \{f_1, f_2, \dots, f_n\}$ .

Assume that  $S$  is *minimally unsatisfiable* (i.e.  $S$  is unsatisfiable, and every *proper* subset of  $S$  is satisfiable, see also definition 3.18).

Is the following assertion true or false?

We can identify  $n$  correct reasonings with premises and conclusions that are formulas in  $S$  or negations of formulas in  $S$ .  $\square$

**META-THEOREM 3.3.**– (interpolation theorem).  $\mathcal{A}, \mathcal{B}$ , and  $\mathcal{C}$  denote wffs of propositional logic.

*Propset* ( $\mathcal{X}$ ): set of propositional symbols in wff  $\mathcal{X}$ .

if:

$$\mathcal{A} \models \mathcal{B} \text{ and}$$

$$\text{Propset}(\mathcal{A}) \cap \text{Propset}(\mathcal{B}) \neq \emptyset$$

then:

there exists  $\mathcal{C}$  such that:

$$\text{Propset}(\mathcal{C}) = \text{Propset}(\mathcal{A}) \cap \text{Propset}(\mathcal{B}) \text{ and}$$

$$\mathcal{A} \models \mathcal{C} \text{ and } \mathcal{C} \models \mathcal{B}.$$

$\mathcal{C}$  is called the interpolant of  $\mathcal{A}$  and  $\mathcal{B}$ .

**EXAMPLE 3.5.**–  $Q \wedge R \models P \vee Q$

$$Q \wedge R \models Q \text{ and } Q \models P \vee Q$$

$$\text{here } \mathcal{A}: Q \wedge R \quad \mathcal{B}: P \vee Q \quad \mathcal{C}: Q \quad \square$$

**EXERCISE 3.18.**–

a) Give the idea of an algorithm that constructs the interpolant and allows us to prove meta-theorem 3.3.

b) Compute the interpolant(s) of:

$$A \Leftrightarrow (B \vee C) \models (A \wedge \neg B) \Rightarrow C.$$

Compute the interpolant(s) of:

$$(\neg A \wedge \neg B) \wedge (A \Leftrightarrow C) \models (C \Rightarrow B) \wedge (\neg D \vee \neg C). \quad \square$$

### 3.2. The method of semantic tableaux

We have characterized a correct reasoning (from a semantical point of view) as a reasoning in which *there is no interpretation that evaluates the premises to **true** and the conclusion to **false*** (simultaneously). As a consequence, if we try to construct all possible models of the set of premises and of the *negation* of the conclusion, then we are “killing two birds with one stone”. Indeed, if we do not succeed, then we were

unable to refute the reasoning, because we were unable to evaluate the premises to **true** and the conclusion to **false** (as we tried to evaluate the negation of the conclusion to **true**). Hence, the reasoning is correct. However, if we succeed, then we will have produced a counter example that proves the reasoning is not correct.

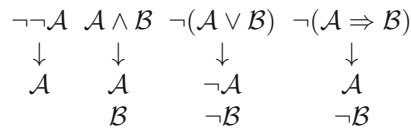
We also noticed that it is not always necessary to construct an entire truth table to evaluate a formula to **true** (see exercise 3.5).

These ideas form the basis of a method that is used not only in a propositional logic but that also extends to first-order logic, to modal logics, etc.

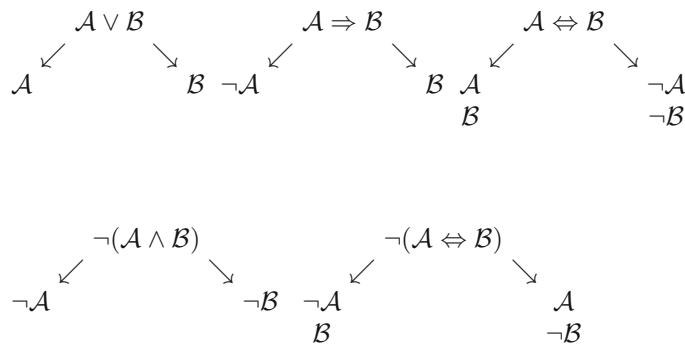
The method of semantic tableaux, of which many variants are available, is a formalization of these ideas. It is based on the following technique: for every connective (formula constructor) that can occur in a wff  $\mathcal{A}$ , we provide all the ways of constructing models of  $\mathcal{A}$ . Of course, these possibilities are encoded in the simplest possible way.

The set of connectives that are used is  $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ , and the method uses the following rules ( $\mathcal{A}$  and  $\mathcal{B}$  denote wffs):

Type  $\alpha$  rules (wffs with only one descendant)



Type  $\beta$  rules (wffs with two descendants)



Intuition behind the method: given a set  $S$  of wffs and using the rules above, we try to construct all the models of  $S$  by combining the models of each formula.

It is clear that the rules replace wffs by wffs containing (strictly) less symbols; thus, we necessarily reach literals (this property is used to prove the termination of the method).

The combination of models translates into a tree (which is not unique and depends on the order in which the models of  $S$  are analyzed).

Some combinations may not be possible (because we are trying to evaluate an atomic formula to **true** to construct the model of one wff, while evaluating it to **false** to construct the model of another one). As soon as a branch contains literals  $A$  and  $\neg A$ , it is closed (to close a branch, we will use the symbol  $\times$ ). We shall say that the branch is *closed*. If a branch is not closed, then it is *open*.

A tableau is *closed* iff all its branches are closed. It is *open* otherwise.

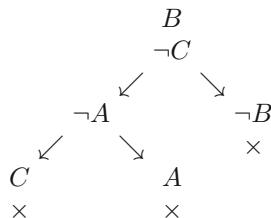
Before providing the corresponding algorithm, we apply the rules on a few simple examples.

In agreement with what we said at the start of the section, we consider the set (which means that *there is no imposed ordering* to analyze the formulas) of formulas:

- 1)  $A \Rightarrow \neg B$
- 2)  $\neg C \Rightarrow A$
- 3)  $\neg(B \Rightarrow C)$

We will try to construct a (several) model(s) for the set of formulas (1), (2), and (3). The order selected for the analysis is (3), (1), and (2). The procedure is:

completely mechanical: for each wff, we identify the rule to apply  
(there is one for each connective).  
We do not try any simplifying transformation.



*Conclusion:* All the branches in the tree are closed. This means that we have failed to construct any model of (1), (2), and (3). This set of wffs is therefore

contradictory (or unsatisfiable, or incoherent). The original reasoning was correct (see exercise 3.15).

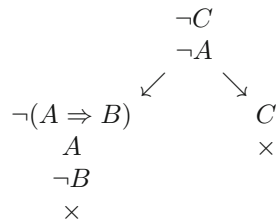
Consider the following reasoning:

$$\frac{(A \Rightarrow B) \Rightarrow C}{\neg C \Rightarrow A}$$

We analyze the set:

- 1)  $(A \Rightarrow B) \Rightarrow C$
- 2)  $\neg(\neg C \Rightarrow A)$

We choose to apply the rules in the order (2), (1).



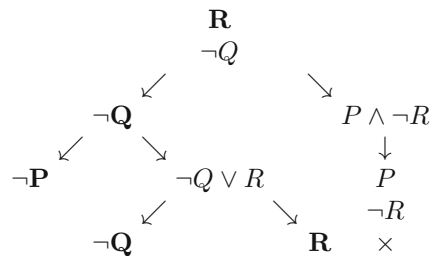
We have analyzed two examples of correct reasonings. What happens when a reasoning is not correct? Is the method also useful? Consider the following reasoning:

$$\frac{P \Rightarrow (\neg Q \vee R) \quad Q \Rightarrow (P \wedge \neg R)}{R \Rightarrow Q}$$

We therefore consider the following set of wffs:

- 1)  $P \Rightarrow (\neg Q \vee R)$
- 2)  $Q \Rightarrow (P \wedge \neg R)$
- 3)  $\neg(R \Rightarrow Q)$

and we choose to analyze it in the order (3), (2), and (1).



As we are trying to enumerate all the potential models of the set of wffs under consideration, in principle, we should have grafted the subtree with root  $\neg Q$  instead of  $\times$ , but of course, this is not necessary because all the branches starting at this node will contain the contradiction  $(R, \neg R)$ .

What do open branches mean? By reading the atomic wffs and the negations of atomic wffs, we obtain the models of the premises and of the negation of the conclusion; hence the models of the premises are counter models of the conclusion, i.e. counter examples showing that the reasoning is not correct.

The counter examples that we can read on the open branches are:

$$\{R, \neg Q, \neg P\}$$

$$\{\neg Q, R\}$$

REMARK 3.10.– It should be clear that the method of semantic tableaux for PL (PC) is a method for enumerating the models of a set of formulas, and thus permits us to decide the (un)satisfiability ((in)coherence) of a set of wffs.  $\square$

EXERCISE 3.19.– Use the method of semantic tableaux to prove that the set of formulas below is unsatisfiable (contradictory or incoherent). Use two different analysis orderings to show that the size of the tree depends on the ordering.

- 1)  $\neg((P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q))$
- 2)  $P \wedge (Q \Rightarrow (R \vee S))$
- 3)  $\neg(P \vee Q)$
- 4)  $P$
- 5)  $Q \Rightarrow (R \vee S)$   $\square$

### 3.2.1. A slightly different formalism: signed tableaux

This formalism will be useful for logics other than classical logic, which we are currently studying. It speaks sufficiently for itself to require any explanation.

We want to prove the validity of the following wff (i.e. prove that it is a tautology: it is evaluated to **true** in every interpretation):

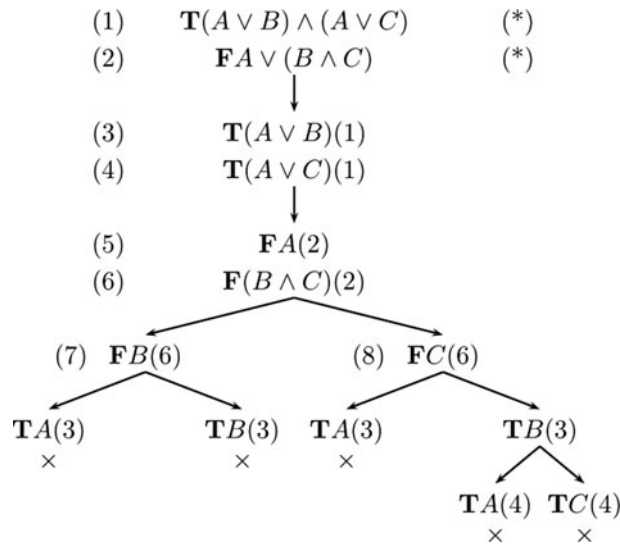
$$(A \vee B) \wedge (A \vee C) \Rightarrow A \vee (B \wedge C)$$

we therefore consider the wff:

$$(*) \quad \neg[(A \vee B) \wedge (A \vee C) \Rightarrow A \vee (B \wedge C)]$$

which leads to the following tableaux:





REMARK 3.11.— This formalism may seem redundant in bi-valued logic, its utility will become obvious in multi-valued logics (see section 10.1). □

Normally, several questions arise at this point of the presentation of the method of semantic tableaux, to which answers must be provided in a systematic study of the topic. The order of the questions is not related to their importance.

- 1) Must the conclusion (or the formula if there is only one) be negated to test its validity?
- 2) Are the constructed trees always finite?
- 3) What is the termination criterion?
- 4) When the method states that a reasoning is correct, is this really the case? (If so, the method is *correct*).
- 5) Can all correct reasonings be detected with this method? (If so, the method is *complete*).
- 6) Is the order in which the formulas (or sub-formulas) are analyzed important?

We begin by answering YES to question 1.

The method of semantic tableaux was created to enumerate all models of a set of formulas (and in particular, of sets containing only one formula).

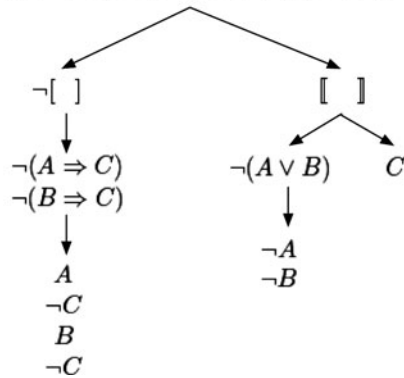
By definition, if  $S$  is a set of wffs:

- set of models of  $S \subseteq$  set of interpretations of  $S$ ;
- set of (all the) valid wffs  $\subset$  set of (all the) non-contradictory wffs.

If no negation is performed, then all models are enumerated, but we do not know anything (without some additional reasoning) about the potential counter examples. Consider the following wff.

EXAMPLE 3.6.-

$$\mathcal{F} : [(A \Rightarrow C) \vee (B \Rightarrow C)] \Rightarrow [(A \vee B) \Rightarrow C]$$

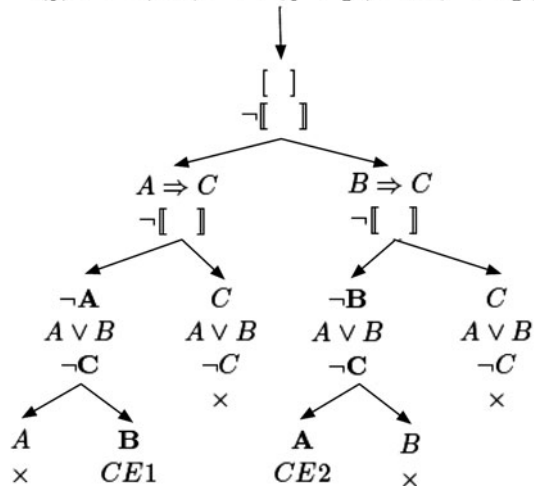


We have proved that  $\mathcal{F}$  is not contradictory, but we do not know whether it is valid.  $\square$

If we negate, we try to construct all models of  $\neg\mathcal{F}$ , i.e. all counter models (counter examples) of  $\mathcal{F}$ .

EXAMPLE 3.7.-

$$\neg([(A \Rightarrow C) \vee (B \Rightarrow C)] \Rightarrow [(A \vee B) \Rightarrow C])$$



$\square$

```

algorithm SEMANTIC TABLEAUX (PL)
% constructs the tree
input: a finite set of PL formulas  $\mathcal{F} = \{f_1, \dots, f_n\}$ 
output: a set of models of  $\mathcal{F}$  or  $\mathcal{F}$  contradictory
start
  root of the tree  $\leftarrow \mathcal{F}$ 
  while  $\mathcal{F} \neq \emptyset$  and (there remain open branches)
  do
    if a branch  $\mathcal{B}$  contains 2 complementary literals, close  $\mathcal{B}$  (put symbol  $\times$ )
    % the corresponding model is not viable
    choose  $f_i \in \mathcal{F}$  and apply the corresponding rule that produces (say)
     $f_i^j$  ( $1 \leq j \leq 2$ )
    Graft  $f_i^j$  ( $1 \leq j \leq 2$ ) to all open
    branches passing through the node labelled by  $f_i$ 
      if  $f_i^j$  ( $1 \leq j \leq 2$ ) is not a literal
      then  $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\}) \cup \{f_i^j\}$ 
      else  $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\})$ 
    endif

  if the tree is closed
  then return  $\mathcal{F}$  contradictory
  else return the sets of literals of the open branches
end

```

**Figure 3.1.** The semantic tableaux (PL) algorithm

We have constructed two models of  $\neg\mathcal{F}$  (*CE1* and *CE2*). *CE1* ( $\{\neg A, \neg C, B\}$ ) and *CE2* ( $\{\neg B, \neg C, A\}$ ) are counter models (counter examples) of  $\mathcal{F}$ , which is therefore *not valid*, as we might have believed after a hasty interpretation of the tableaux in which  $\mathcal{F}$  was not negated.

REMARK 3.12.– We have indirectly answered questions 1, 2, 3, and 6 given earlier. The proposed algorithm is *non-deterministic* (instruction **choose**) and can therefore be implemented with different *strategies*.  $\square$

EXERCISE 3.20.– Use the method of semantic tableaux to verify the (in)correctness of reasonings (a) to (f), and the (in)validity of formula (g) below.

a)

$$\frac{\begin{array}{l} \neg H \Rightarrow M \\ W \Rightarrow \neg H \end{array}}{\neg H \Rightarrow (\neg W \Rightarrow M)}$$

**How should the method be used?**

– To test the correctness of a reasoning  $H_1, \dots, H_n \models^? C$ , take  $\mathcal{F} \leftarrow \{H_1, \dots, H_n, \neg C\}$ . If the algorithm answers  $\mathcal{F}$  **contradictory**, then the reasoning is correct, otherwise the algorithm provides all the counter examples (open branches) that prove that the reasoning is incorrect.

– To prove that a formula  $\mathcal{G}$  is valid, take  $\mathcal{F} \leftarrow \{\neg \mathcal{G}\}$ . If the algorithm answers  $\mathcal{F}$  **contradictory**, then the formula is valid, otherwise the algorithm provides all the counter examples that prove the formula is not valid.

– To construct all the models of a formula  $\mathcal{G}$ , take  $\mathcal{F} \leftarrow \{\mathcal{G}\}$ .

– To prove that a set of wffs  $\mathcal{S}$  is contradictory (unsatisfiable), take  $\mathcal{F} \leftarrow \mathcal{S}$ . If the tree is closed then  $\mathcal{S}$  is unsatisfiable, otherwise we obtain all the models of  $\mathcal{S}$ .

– To test whether a set of wffs  $\mathcal{S} = \{f_1 \dots f_n\}$  is valid, take  $\mathcal{F} \leftarrow \{\neg f_1 \vee \dots \vee \neg f_n\}$ . If the tree is closed then  $\mathcal{S}$  is valid, otherwise we obtain all the counter examples of  $\mathcal{S}$ .

b)

$$\frac{(R \Rightarrow G) \wedge (\neg R \Rightarrow S)}{G \vee S}$$

c)

$$\frac{(H \vee W) \Rightarrow M}{M \vee H}$$

d)

$$\frac{\begin{array}{l} A \Rightarrow (\neg B \vee C) \\ B \Rightarrow (A \vee \neg C) \end{array}}{C \Rightarrow B}$$

e)

$$\frac{\begin{array}{l} P \Rightarrow (R \wedge T) \\ (T \vee S) \Rightarrow \neg Q \end{array}}{\neg(P \vee Q)}$$

f)

$$\frac{\begin{array}{l} (S \Rightarrow R) \wedge P \\ Q \end{array}}{\neg R \wedge (\neg S \wedge P)}$$

g)

$$[(P \wedge Q \wedge R) \Rightarrow S] \Leftrightarrow [[P \Rightarrow (Q \Rightarrow (R \Rightarrow S))]] \quad \square$$

EXERCISE 3.21.– Are the sets of wffs  $\mathcal{S}_1$  and  $\mathcal{S}_2$  below satisfiable? Unsatisfiable?

Use the method of semantic tableaux to find the answer.

a)  $\mathcal{S}_1 = \{P \Rightarrow Q, R \Rightarrow S, \neg(\neg P \wedge \neg R), \neg(P \wedge S), R \Rightarrow \neg Q\}$

b)  $\mathcal{S}_2 = \{\neg P, \neg R \Rightarrow W, Q \vee (\neg T \Rightarrow \neg(P \vee U)), \neg P \Rightarrow (U \wedge \neg R), \neg Q, \neg U, \neg T, \neg R \Rightarrow S\}$ .  $\square$

DIGRESSION 3.1.– (some refreshers on trees).

– A directed graph is a structure (see definition 5.4)  $\mathcal{G} = \langle G; R^{\mathcal{G}} \rangle$ , where  $R^{\mathcal{G}} \subseteq G^2$  is an irreflexive relation<sup>3</sup>.

$G$ : set of nodes;  $R^{\mathcal{G}}$ : set of edges.

– A tree is a directed graph with a (unique) distinguished node  $\mathbf{r}$ , called its root, such that:

i) no edge reaches  $\mathbf{r}$  (i.e.  $\forall x(x, \mathbf{r}) \notin R^{\mathcal{G}}$ );

ii) for all nodes  $n$  in the tree, there exists a unique path (branch) from  $\mathbf{r}$  to  $n$ .

– If there is a path from a node  $x$  to a node  $y$ , then  $y$  is a descendant of  $x$  and  $x$  is an ancestor of  $y$ .

– The degree of a node is the number of outgoing edges of the node.

– A node of degree 0 is called a terminal node, or a leaf.

– The length of a branch is the size of the set of its nodes.

– A tree is infinite iff it has infinitely many nodes.

– A tree is finitely generated iff all its nodes have a finite degree.

REMARK 3.13.– (other definitions of a tree). A tree is a partially ordered set (with order relation  $\prec$ , see definition 3.24) satisfying the additional axiom:

$$\forall x \forall y \forall z (y \prec x \wedge z \prec x \Rightarrow y \prec z \vee z \prec y)$$

(this means that the set of minorants of each element is totally ordered).  $\square$

THEOREM 3.1.– (König's lemma). *If  $A$  is an infinite tree that is finitely generated, then  $A$  contains (at least) an infinite branch.*

---

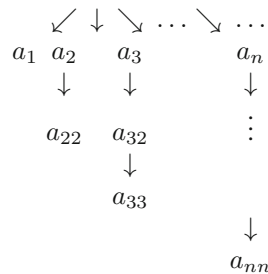
<sup>3</sup> Irreflexivity ( $\forall x \neg R^{\mathcal{G}}(x, x)$ )  $\neq$  non-reflexive ( $\exists x \neg R^{\mathcal{G}}(x, x)$ ).

or

If  $A$  is a tree with only finite branches then  $A$  contains a branch of maximal length.

or If for all  $n \in \mathbb{N}$  there exists a node of depth  $n$ , then  $A$  contains an infinite branch.

König's lemma does not apply to trees that are not finitely generated, as shown in the example below:



### 3.3. Formal systems

The notion of a formal system is closely related to the notion of proof.

#### 3.3.1. A capital notion: the notion of proof

The notion of a *proof* is extremely important and subtle. It is thus very hard (and perhaps impossible) to grasp its meaning by simply relying on a formal definition. It occurs in many domains, in exact science (mathematics, physics, biology, chemistry, etc.), in social science (history, paleontology, sociology, etc.), and in many other activities that are proper to organized societies (justice, police, sales, insurance, etc.).

Etymologically:

proof  $\rightarrow$  to prove  $\rightarrow$  from *probus*: of a good quality, honest, loyal

The requirements to identify an object as a proof and the methods to carry out proofs are of course generally very different and depend on the period under consideration (*including for mathematics*).

For example, it is currently acknowledged in the so-called civilized societies that every individual is innocent until *proven* guilty.

This was not the case in the code of Hammurabi (King of Babylon, ~1750 BC). The defendant would be thrown into the “holy river”: if he could escape unhurt, that was a *proof* that he was innocent; if he sank, that was a *proof* that he was guilty. This proof method, along with other torture methods, were used by the Inquisition during the Middle Ages.

Myths used to be *considered as true*, and anyone who would doubt them had to *prove* the contrary.

In mathematics, the acceptance of what a proof is can also be analyzed from a historical point of view. For example, pre-Euclidean proofs were much less rigorous than post-Euclidean ones. A similar remark holds for mathematical analysis before and after Weierstrass. It seems like all societies, including those considered as primitive, invented argument criteria, often as questions/answers, in particular in the domains of law and justice. Notions such as coherence are often used (but not necessarily in a conscious way) in the evaluation of arguments. An argument that tends to establish a fact (or situation) that has not been observed or to reduce the uncertainties about this situation can already be considered as *proof*. Once again, those who pushed the study of argument the farthest were the Greeks, in particular, between the 6th and 4th Century BC, with major works such as Aristotle’s *Organon* and *Rhetoric*, and Euclid’s *Elements*. The accomplishment that we are particularly interested in is the development of *axiomatizations* and *proofs*.

There were abstract arguments, before Aristotle, with, for example, Thales, Anaximander, Xenophanes, Heraclitus, and Parmenides. People already used *reductio ad absurdum*, analogies, *dilemmas* (if  $p \Rightarrow q$  and  $r \Rightarrow s$  and  $p \vee r$ , then  $q \vee s$  is a dilemma. If  $p \Rightarrow q$  and  $r \Rightarrow s$  and  $\neg q \vee \neg s$ , then  $\neg p \vee \neg r$  is another one).

There seems to be a consensus on the fact that Parmenides was the first to propose proofs with deductive reasoning, with an irrefutable initial statement and a rigorous chain of deductions. He also proposed to divide proofs into parts, using the conclusions of previous parts as premises of the current ones (today these would be called *lemma-structured proofs*).

The influence of these thinkers on the development of philosophy and science was huge. To better understand the social context in which the notion of a proof evolved toward its current definition, it is worth mentioning that argument blossoms much better and has a greater importance in a society in which the opinion of the citizens (who are perfectly equal) is taken into account (as was the case in Greece at the time<sup>4</sup>) than in an autocratic society.

---

<sup>4</sup> The Greeks invented democracy at the same time.

Argument always takes place between two people (this is not always the case for reasoning).

Sophists, and in particular Gorgias, used the reasonings that were introduced by Parmenides and his students, not to find the truth but with a purely rhetorical goal. They believed *persuasion* was much more important than a proof, although the latter aims at truth and knowledge. Still, it is now acknowledged that rhetoric indirectly contributed to the progress in the concept of a proof.

Sophists also played an important role in the organization of education and in the development of a critical mind, by admitting several points of view on a same topic<sup>5</sup>. Socrates, who used to search for general definitions through dialog, could profit from the habits of Athenians for his teachings.

Socrates defined the art of rhetoric as *the art of having influence on souls*. Aristotle considered rhetoric as the technique of *plausibility*, but not of truth (of which proofs are the technique).

Plato must not be forgotten among those who contributed to the elaboration of the concept of proof.

Another discipline that is close to rhetoric is *dialectic*<sup>6</sup> (instrument of probable knowledge according to Aristotle), which, according to Aristotle, is due to Zeno of Elea, was also important at the time.

It is interesting to notice that the clear separation between proving and persuading (convincing) disappears a little in philosophy and modern mathematics (see below).

Mathematics are of course the principle domain in which proofs are carried out. In a reference book (by Rasiowa and Sokorski), it is written:

The process of deducing some sentences from others is the most important part of the mathematical work.

The more common belief is that:

The first proof in the history of mathematics was produced by Thales of Miletus (600 BC): “A diameter separates a circle into two equal parts”.

As suggested by the brief historical recollection below, this notion *evolved* with time and it is legitimate to assume that Thales used simple and intuitive techniques.

---

<sup>5</sup> Rhetoric was part of the French official education program until the 19th Century. There was a renewed interest in rhetoric during the 20th Century.

<sup>6</sup> Dialectic comes from a verb meaning to discuss.



The priority given to Thales does not seem well deserved. Indeed, one of the greatest historians of mathematics (E.T. Bell) wrote the following about Babylonian mathematics:

...More important than the technical algebra of these ancient Babylonians is their recognition – as shown by their work – of the necessity of *proof* in mathematics.

Until recently, it has been supposed that the Greeks were the first to recognise that proof is demanded for mathematical propositions. This was one of the most important steps ever taken by human beings. Unfortunately, it was taken so long ago that it led to nowhere in particular so far as our own civilisation is concerned, unless the Greeks followed consciously, which they may well have done. They were not particularly generous to their predecessors.

The first rigorous mathematical proofs were produced during the 5th Century BC. The concept of a *rigorous mathematical proof* is certainly the main difference between Greek mathematics and Babylonian mathematics.

It is worth mentioning that the ancient Greeks made a distinction between *finding* a theorem and *finding its proof*. We know that many of Euclid's propositions were already known and accepted before his time.

The main development, of which Euclid's monumental work is the paradigm, is that of an axiomatization.

After Euclid, mathematical reasonings used *definitions*, *common notions*, and *postulates*.

– *Common notions* are the obvious principles that apply everywhere in mathematics. They are now called *logical axioms*. Common notions seem to contain the laws of logic (with the current meaning of *inference rules*).

– *Postulates* are the basic hypotheses on geometry (or another theory). They are now called *proper axioms* or *non-logical axioms*.

Some authors have suggested that instead of *axiomatization*, we should use *postulation*.

DIGRESSION 3.2.– The way Euclid proceeded in his work has become a standard in mathematics and other domains, among which is medicine, which was one of the first domains to incorporate it.

Galen (physician, surgeon, and philosopher of the 2nd Century) believed that mastering proofs was a prerequisite to study medicine, and he thought that everything that is proved by medical science should be reduced to prime propositions that are

unprovable but held as true by everyone. This boils down to importing Euclid's axiomatization method to medicine.

Recent research by science historians has highlighted an approach to proofs that is particularly interesting to computer scientists. Chinese mathematicians (1st Century BC) had proposed (sometimes sophisticated) *algorithms* to solve problems. These algorithms would provide solutions to many particular cases of the problems under consideration. Those commenting them tried to prove the veracity of the propositions, which boiled down to proving the correction of the algorithms.

At the origins of modern science (16th to 17th Century), knowledge was associated to sensitive experiments (that, just as for theories, had to be transmissible and reproducible<sup>7</sup>).

This is different from revelation, illumination, initiation, and hermetism as a means of discovering and transmitting knowledge.

In one of his books, Galileo mentions the *imperfections of matter* and the *very pure mathematical proofs*.

Kepler was convinced that mathematical proofs were the way to reach the truth.

Newton (following Euclid) uses the axiomatic method: he starts from the definition of mass, force, and movement. Then, he adds the presuppositions, laws, axioms. He obtains theorems, proofs, and corollaries. To get from abstract entities to a description of the world, Newton states philosophical rules. □

Some mathematicians have put an emphasis on the fact that *from a practical point of view*, the acceptance of the proof of a theorem is a *social process*. There are interactions between those who propose a proof and those who verify it, detect possible errors, simplify it, make it readable, etc. The process stops when the community of mathematicians accepts (or refutes) the (alleged) proof.

Much later, the appearance of computers led to hopes and new problems about the notion of a proof. Indeed, what is more tempting than trying to prove theorems using a computer? All we have to do is to program a notion that seems completely formalized, and this led to what is considered as the first AI program: *Logic Theorist* (see section 7.5).

The first theorem prover that produced a mathematical proof was written by M. Davis in 1954. It was a decision procedure for Presburger arithmetic (see

---

<sup>7</sup> See below the characteristics of a proof when answering the question “What is a proof?”.

example 5.6). It is significant that the first problem to be tackled was a *decidable* one (of a high complexity): computers were used as large calculators.

However, it is only with the proof of the *four-color theorem* that mathematicians and the general public started talking about automated deduction.

The four-color theorem was proved using a computer program in 1976. This result is interesting for several reasons: it had been an open problem for over a century in mathematics, it had allegedly been proved by some excellent mathematicians who had produced (false) proofs (in the traditional way, i.e. without a computer), and it permitted some interesting thoughts on the notion of a proof in mathematics. This result, as well as others that followed, essentially uses the computing speed of a computer to test a huge amount of possible cases. Although the result is very important, the proof only made a very partial use of the capacities of automated theorem provers, as they are designed nowadays (in particular, no use was made of the possibilities of manipulating proofs, planning, interacting with the user to guide parts of the proofs, logical computations, etc., that are offered by modern theorem provers). It is interesting to note here the important consequences that these results had on mathematical philosophers, as an inspiration for their thoughts on mathematical practices.

The proofs that are obtained by a computer make an argument that the acceptance of a proof is a social process. The main reasons are the large number of computations that are performed (without any creativity), the lack of a perspective in the proofs (that do not distinguish those steps that are conceptually important from the other ones), and also the natural mistrust of humans toward a proof that was produced by a non-human. Let us not forget that in most cases, we “trust” competent mathematicians when they say that some assertions have been proved. It suffices, for example, to recall Fermat’s last theorem.

In a reference article (by T. Tymoczko) about the implications of this work on the philosophy of mathematics, the author proposes the following thesis (that seems daring at first):

I will argue that computer-assisted proofs introduce experimental methods into pure mathematics.

To the question “What is a proof?”, the author answers by identifying three main characteristics:

– Proofs are *convincing*<sup>8</sup>

---

<sup>8</sup> Here, the author’s requirements coincide with those that form the basis of proof theory.

- Proofs are *surveyable*<sup>9,10</sup>.
- Proofs can be formalized<sup>11</sup>.

Surveyability is an important subjective feature of mathematical proofs which relates the proofs to the mathematicians, the subjects of mathematical investigations. It is in the context of surveyability that the idea of “lemma” fits. Mathematicians organize a proof into lemmas to make it more perspicuous.

Other authors have almost pushed this analysis to its limit: they maintain that mathematical studies have an important empirical component (and therefore inherit its methods: reproducibility of the experiments, etc.<sup>12</sup>).

It is interesting to compare the theses given above with the thoughts that inspired a *mathematician* (C.W.H. Lam) in his own work, as he obtained new results (in 1977) using a computer. He begins his article by explaining his work, using the title of an article that had appeared in a general magazine:

Is a math proof a proof if no one can check it?

The proof involved the projective plans of order 10 and required 3000 CPU hours to a CRAY-1A, for which scientists used to believe that there were undetected (material) errors every 1000 hours approximately...!

He tries to use the expression *computed result* instead of “proof” and states that in the case of the proofs obtained by a computer, the assertion of correctness is not *absolute*, but only *almost certain*, which is a characteristic of a *computer-based* result.<sup>13</sup>

This kind of problem is in a close relationship to others that are clearly related to practical computer science.

---

9 Of course, this requirement also holds for other human activities: a writer noticed that since 1922 and until the correct edition, more than 5,000 printing errors had been made in Joyce’s *Ulysses*. Because the book was believed to be incomprehensible, no one had noticed the mistakes.

10 There should be a special mention to *probabilistic* proofs and *zero-knowledge* proofs. In the former, random verifications with negligible possible errors can be carried out. In the latter, someone who does not know of a proof produced elsewhere can be convinced of the correctness of a result without knowing how it was obtained.

11 This is clearly related to the idea behind *proof assistants* and *logical frameworks*.

12 There exists a mathematical journal named *Experimental Mathematics*.

13 The author forgot that there were many wrong “proofs” (obtained by excellent mathematicians), like those mentioned in the four-color theorem, well before computers ever existed.

For example, when we carry out a *proof* or a *verification* (e.g. for a critical system), we prove that a program is correct, which means that it will do what is expected from it on a *model of a computer*, but have we proved that it will do it on a *real* computer (that has a physical reality, with electronic components, etc.)?

On this topic, the great logician P. Martin-Löf (who influenced computer science a lot) wrote:

Now, there can be no proof, no conclusive proof, that a proof really proves its conclusion, because, if such a miraculous means were available to us, we would of course always use it and be guaranteed against mistakes. It would be a wonderful situation, but we are human beings and we are not in that position

[ ... ]

So, clearly, there can be no proof in the proper sense of the word that a proof really proves its conclusion: the most there can be is a discussion as to whether a proof is correct or not, and such discussion arises precisely when we make mistakes, or when we have insufficient grounds or evidence for what we are doing.

The importance of the presentation (readability) of a proof (“proofs are surveyable”) cannot be overestimated:

Every result that is obtained by a means that is not surveyable by a human is not a proof.

This sentence was written by a great French mathematician (J.-P. Serre, recipient of the Fields and Abel medals).

Actually, a response to this requirement, which is not in contradiction with the acceptance of proofs obtained using a computer, is to make the following distinction along with the logician quoted above: there are *proofs* and there are *derivations*. A *proof* contains the computational information that a computer would need to verify a proposition. A *derivation* is what *convinc*es us that a proposition is true. *Derivations* are what we find in mathematics textbooks. These considerations are closely related to what the same logician wrote in a brilliant article:

... Thus proof and evidence are the same. And what is it that makes a judgement evident to you? Before you have understood or grasped the judgement, it is not evident to you, and, when you have grasped it, it is obvious or evident to you. This is simply your act of understanding or grasping it which confers evidence on the judgement, that is, which makes it evident to you. ...

What is of a particular interest to us here is the word *grasped*. We cannot grasp (with our mind) extremely long sequences of symbols without any structure.

As a great logician (Y. Manin) nicely put it: “A good proof is a proof that makes us wiser”.

To conclude, recall that natural science relies (and depends on) many instruments (it suffices to consider astronomy or biology). The history of instruments, in which computers play an important part, is part of the history of science.

It is a commonplace to say that computers are indispensable for complicated numerical computations. They will probably be as indispensable as *reasoning* auxiliaries, to obtain (some) proofs.

### 3.3.2. *What do we learn from the way we do mathematics?*

After this brief historical perspective, we can try to rely on our direct experience of mathematics (although it is very modest) to better comprehend the topic and ask ourselves, for example:

Can we abstract common characteristics of the proofs we have discovered, read, studied, etc.?

Proofs are presented as a finite sequence of formulas, sometimes with figures (that correspond to particular cases: examples (models), counter examples (counter models)), and with sentences in natural language (generally a very restricted subset of the everyday language) that justify the introduction of new formulas, and... that's it!

– What formulas do we begin with?: by the “unquestionable” formulas, which are admitted.

– How do we get from some formulas to others?: by some rules, generally not many of them (in general we do not bother to specify that they are the only ones we allow ourselves to use), that are implicitly correct and assumed to be natural.

We have just given the basic ideas of what we shall define as a *formal system*.

– The unquestionable formulas are the *axioms*.

– The transition rules are the *inference rules*.

To avoid any artificial problem (such as ambiguity), we fix a *formal* language.

The characterization of inference rules is more delicate than the characterization of axioms. Here are some of the most fundamental characterizations:

1) The hypothetical syllogism (*modus ponendo ponens*) or simply *modus ponens*:  
from  $A$  and if  $A$  then  $B$  deduce  $B$ ;

2) Induction: the great mathematician Henri Poincaré (1854–1912) who was also a philosopher of science, considers induction as the fundamental mathematical reasoning tool and states that its essential character is that it contains *infinitely many* hypothetical syllogisms:

The theorem is true for 1

(\*) But if it is true for 1 then it is also true for 2.

Therefore, it is true for 2.

(\*) But if it is true for 2 then it is also true for 3.

Therefore, it is true for 3.

...

Furthermore, there is a unique formula that expresses *all* the formulas (\*):

(\*) If the theorem is true for  $n - 1$ , then it is also true for  $n$ .

The principle of mathematical induction seems to have been used in its current form by B. Pascal in 1654 in the *Traité du triangle arithmétique*.

From the history of science point of view, it is interesting to note that al-Karaji ((Persian) mathematician and engineer, 953 to ~1029) used a rudimentary form of mathematical induction, by proving an argument for  $n = 1$ , then using this result to prove the result for  $n = 2, \dots$ , and noticing that we could go on indefinitely. He discovered what is known as “Pascal’s triangle”, using this method.

It seems like Pascal was not aware of al-Karaji’s work.

3) Sometimes, we use *modus tollendo tollens* (or simply *modus tollens*):

from *if A then B* and  $\neg B$  deduce  $\neg A$ , which can be considered as a particular case of *reductio ad absurdum* (see below).

Three widely used *techniques* to carry out proofs are as follows:

t1) *Reductio ad absurdum*.

This is one of the oldest techniques. There are two cases to consider.

a) When it is used to prove that an object exists, it is closely related to the law of excluded middle. To prove  $P$ , we prove that  $\neg P$  leads to a contradiction. By the law of excluded middle,  $P \vee \neg P$  is always true; hence, we may conclude that  $P$  holds. Intuitionists do not always accept these proofs because the law of excluded middle is used;

b) When it is used to prove that a mathematical object does not exist, it is accepted by all schools of thought. If  $P$  leads to a contradiction, then the object having property  $P$  cannot exist (without any other consideration).

t2) Case analysis, particularly important for proofs performed by computers, such as the proof of the four-color theorem and the projective plane of order 10.

t3) Diagonalization. This procedure was invented by Cantor and is used to perform proofs by *reductio ad absurdum*; we assume that we can enumerate all the objects of a class and the diagonalization procedure constructs an element of the class that is not among the enumerated objects. Assuming that there exists such an enumeration therefore leads to a contradiction (see exercise 3.1).

REMARK 3.14.– It might be useful to recall that three different theories must be distinguished:

- the *informal theory*, that the formal system is meant to formalize;
- the formal system or *object theory*;
- the *meta-theory*, in which the formal system is studied. The meta-theory generally corresponds to common and informal mathematics (see, for example, meta-theorem 3.4). □

REMARK 3.15.– (mathematical theories). Mathematical theories can be considered from the semantic or the syntactic point of view. In the former case, the axioms have an *intended interpretation* (or *wanted interpretation*), and this model is *in principle* unique (see remark 5.19); for example, arithmetic, Euclidean geometry, set theory.

When the syntactic point of view is chosen, we search for those interpretations that satisfy some formulas. In this case, the number of models may be important. Examples: group theory, non-Euclidean geometry.

With the first point of view, the search for an axiomatization is similar to modeling in natural science (see Chapter 5.2).

Of course, both point of views can coexist.

In experimental science, researchers have also defined what can be considered as a *proof* of a scientific theory, which must include (globally) the observation of some facts, the proposal of hypotheses, and a verification (or falsification). See also sections 8.3 and 8.4.

The problems that arise with the notion of a proof in experimental science are extremely difficult (in particular, from an epistemological point of view). □

DEFINITION 3.9.– (formal system). A *formal system or formal theory or axiomatico-deductive system*  $\mathcal{S}$  is a triplet:



$$\mathcal{S} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$$

where:

–  $\mathcal{L}$  is a set of wff.

$\mathcal{L}$  is a formal language on a given vocabulary (assumed to be finite), and it can always be decided in a mechanical way whether a sequence of symbols is a word in the language (i.e. a wff) or not.

–  $\mathcal{R} = \{RI_k \mid k \geq 1\}$  is a finite set of inference rules, i.e. relations in  $\mathcal{L}^n$  ( $n \geq 2$ ).

They are generally denoted as follows:

$$RI_k : \frac{\mathcal{A}_1, \dots, \mathcal{A}_{n-1}}{\mathcal{A}_n}$$

The  $\mathcal{A}_i$ 's ( $1 \leq i \leq n-1$ ) are called the premises,  $\mathcal{A}_n$  the conclusion or direct consequence of  $\mathcal{A}_i$  ( $1 \leq i \leq n-1$ ).

It is possible to mechanically decide whether a wff is a direct consequence of other wffs.

We may accept inference rules without premises, i.e. relations in  $\mathcal{L}^n$  ( $n \geq 1$ ). In this case, the axioms are also inference rules.

The axioms and inference rules are also called transformation rules.

–  $\mathcal{A} \subset \mathcal{L}$  are the axioms;

– the pair  $\mathcal{S} = \langle \mathcal{L}, \mathcal{R} \rangle$  is a deductive system or proof system or calculus;

– the pair  $\mathcal{S} = \langle \mathcal{L}, \mathcal{A} \rangle$  is an axiomatic system or axiomatic structure or axiomatization.

The latter is the most frequently used in mathematics, and  $\mathcal{L}$  is usually not formally specified (a human is supposed to be able to recognize the wffs). There is no restriction on the inference rules that we are allowed to use. These are called informal axiomatic theories and they enable us to obtain informal proofs. In fact, it is possible to prove theorems in group theory or set theory, etc. without having ever studied first-order logic. These proofs can be considered as correct but informal (considering that these kinds of proofs are particularly important in constructive mathematics, see, for example, remark 5.29).

Nevertheless, the importance of formal (and unquestionable) proofs cannot be overstated, as they can be verified by a computer program: a proof assistant (logical frameworks).

REMARK 3.16.– A formal system is only concerned with syntax. With this syntax several meanings or interpretations can be associated. This is evidence of the importance of form in logic.  $\square$

REMARK 3.17.– The principle of non-contradiction and the law of excluded middle (see section 2.2) are not inference rules. They are not properties that hold good for all considered wff.  $\square$

EXAMPLE 3.8.– (arithmetic, Peano's axioms). The set of natural numbers ( $\mathbb{N}$ ) has the following properties:

- 1)  $0 \in \mathbb{N}$
- 2) if  $n \in \mathbb{N}$  then  $s(n) \in \mathbb{N}$ ,    % for all  $n$
- 3)  $0 \neq s(n)$ ,    % for all  $n$
- 4) if  $s(n) = s(m)$  then  $n = m$ ,    % for all  $n, m$

*induction axiom* (see example 9.28):

- 5) let  $P$  denote a property on numbers.

if  $P(0)$  and (if  $P(n)$  then  $P(s(n))$ ) then  $P(x)$  for  $x \in \mathbb{N}$ ,    % for all  $P$

Sometimes the induction axiom is stated as follows:

- 5') if  $S \subseteq \mathbb{N}$  and  $0 \in S$  and (if  $n \in S$  then  $s(n) \in S$ ), then  $S = \mathbb{N}$ ,    % for all  $S$

(see example 9.28).  $\square$

REMARK 3.18.– (on the two statements of the induction axiom). Version 5 is weaker than version 5': as a property can be specified as a finite sequence of words in a language (defined by a grammar), only a denumerably infinite number of properties and natural numbers can be specified, but the set of all subsets of  $\mathbb{N}$  is uncountably infinite (see, e.g. exercise 3.1).

There, therefore, exist theorems on natural numbers that *cannot be proved* using form 5 of the induction axiom.  $\square$

EXERCISE 3.22.– Can you give any reason why inference rules are defined as relations rather than as functions?  $\square$

DEFINITION 3.10.– (provable formula). *The set of provable formulas in a formal system is the smallest set such that:*

- if  $A$  is an axiom, then  $A$  is provable;
- if  $A$  is provable and  $B$  is a direct consequence of  $A$ , then  $B$  is provable;

– if  $A$  and  $B$  are provable and  $C$  is a direct consequence of  $A$  and  $B$ , then  $C$  is provable.

The following definition formalizes the same notion, through those well-known *proofs* and *theorems*.

Etymology:

theorem (16th Century)  $\rightarrow$  theatre  $\rightarrow$  *theôrema*: object of contemplation, of study (what we see, in a show)

A curiosity: empirists (school of medicine, 2nd Century) used to define *theorems* as the knowledge of a thing that has been witnessed a number of times, together with the faculty of distinguishing the opposite event.

In the following definition, if the lines beginning with (#) are included (respectively, excluded) and those beginning with (b) are excluded (respectively, included), we obtain the definition of a proof (respectively, deduction).

DEFINITION 3.11.– (proof, deduction). *Consider:*

$\mathcal{S}$  : a formal system;

$A_i, C$  : wffs (of  $\mathcal{S}$ ).

(b)  $\Gamma$  : set of wffs (of  $\mathcal{S}$ );

$A$  (#) proof of  $C$

(b) deduction of  $C$  from  $\Gamma$

in  $\mathcal{S}$ , is a finite sequence

$A_1, A_2, \dots, A_n$  of wffs such that:

1)  $C = A_n$

2) for  $1 \leq i \leq n$

either:

a)  $A_i$  is an axiom

(b) or  $A_i \in \Gamma$

or:

b)  $A_i$  is a direct consequence by an inference rule from  $A_{i_1}, \dots, A_{i_k}$

$i_j < i$  ( $1 \leq j \leq k$ )

(#)  $C$  is called a theorem and it is denoted by

(#)  $\vdash_S C$  or, if there is no ambiguity, by  $\vdash C$

(b)  $\Gamma$  : set of hypotheses or of premises of the deduction

(b)  $\Gamma \vdash_S C$

(b)  $C$  is a consequence of  $\Gamma$ .

The deduction meta-theorem shall relate these two notions.

REMARK 3.19.– (formal system as an algorithm). Note that a formal system  $\mathcal{S}$  can be considered as an algorithm whose output is the set of theorems of  $\mathcal{S}$ .  $\square$

### 3.4. A formal system for PL (PC)

We will define a formal system that will be denoted by  $\mathcal{S}_1$ .

1)  $\mathcal{L}$

We shall restrict ourselves to wffs that only contain the connectives  $\Rightarrow$  and  $\neg$ . There is no loss of generality, as the other connectives can be replaced using the three following definitions ( $A$ ,  $B$ , and  $C$  denote wffs):

D1)  $A \wedge B \stackrel{\text{def}}{=} \neg(A \Rightarrow \neg B)$

D2)  $A \vee B \stackrel{\text{def}}{=} \neg A \Rightarrow B$

D3)  $A \Leftrightarrow B \stackrel{\text{def}}{=} (A \Rightarrow B) \wedge (B \Rightarrow A)$

2)  $\mathcal{R}$

The only inference rule is *modus ponens* or *law of detachment* (denoted by MP by what follows):

$$MP: \frac{A \quad A \Rightarrow B}{B}$$

(As  $A$  and  $B$  are (meta)variables that denote arbitrary wffs, MP should be called a schema of inference rules.)

3)  $\mathcal{A}$

The set made of the three following axiom schemas.  $A$ ,  $B$ , and  $C$  denote wffs, so that each of the axiom schemas below actually denotes (denumerably) infinitely many wffs.

$$A1) \quad A \Rightarrow (B \Rightarrow A)$$

$$A2) \quad (A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$$

$$A3) \quad (\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B) \quad \square$$

Given the definition of a theorem, it is clear that the set of theorems of  $\mathcal{S}_1$  is denumerably infinite.

REMARK 3.20.– (substitution rule). Some authors explicitly add a *substitution rule* that allows to replace the metavariables by wffs.  $\square$

DIGRESSION 3.3.– (variables).<sup>14</sup> The notion of a variable used here is different from the one used in mathematics and in physics, where a variable simply denotes a quantity (like space, time, etc.) that varies.

Here, variables are symbols that can be replaced by expressions from different syntactical categories.

Logic historians agree on the fact that variables were introduced by Aristotle. Since then they have been used by logicians and mathematicians.

Aristotle would use letters as signs that denote “holes”, that can be filled by arbitrary terms, with the constraint that “holes” denoted by the same letter must be replaced by the same term. This technique was of course a major breakthrough in logic, and it is indispensable for the specification of rules such as syllogisms.  $\square$

REMARK 3.21.– In the following pages (respectively, in the solutions to the exercises),  $A_1$ ,  $A_2$ ,  $A_3$  (respectively,  $A_1$ ,  $A_2$ ,  $A_3$ ) denote the axiom schemas of  $\mathcal{S}_1$ .  $\square$

EXAMPLE 3.9.– We show that:

$$\vdash_{\mathcal{S}_1} A \Rightarrow A$$

---

<sup>14</sup> See also digressions 5.2 and 9.1.

Here is a proof:

$$1) (A \Rightarrow ((A \Rightarrow A) \Rightarrow A)) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$$

$$\text{in (A2)} \quad B \leftarrow A \Rightarrow A \quad ; C \leftarrow A$$

$$2) A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$$

$$\text{in (A1)} \quad B \leftarrow A \Rightarrow A$$

$$3) (A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A)$$

**1, 2, MP**

$$4) A \Rightarrow (A \Rightarrow A)$$

$$\text{in (A1)} \quad B \leftarrow A$$

$$5) A \Rightarrow A$$

**3, 4, MP**

Here is another one:

$$1) (A \Rightarrow (B \Rightarrow A)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow A))$$

$$\text{in (A2)} \quad C \leftarrow A$$

$$2) (A \Rightarrow B) \Rightarrow (A \Rightarrow A)$$

**(A1), 1 and MP**

$$3) (A \Rightarrow (B \Rightarrow A)) \Rightarrow (A \Rightarrow A)$$

$$\text{in (2)} \quad B \leftarrow B \Rightarrow A$$

$$4) A \Rightarrow A$$

**(A1), 3 and MP**

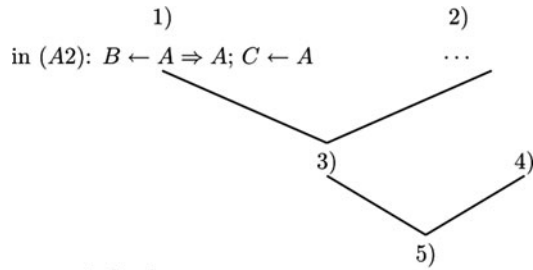
□

REMARK 3.22.– (A3) was not used in any of these proofs.

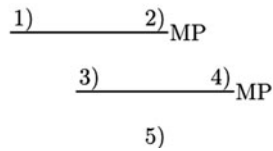
□

DIGRESSION 3.4.– The *linear* representation of proofs, which is the one we shall adopt, is not fundamental. The proofs of example 3.9 could have been represented in a tree-like manner (for the first one).

For the sake of readability, we only give the numbers that identify the formulas.



or equivalently:



□

A few thoughts: does it seem possible to write an algorithm that *verifies* that a given sequence of wffs is a proof in a formal system?

In the case of a positive answer, what are the main problems that arise?

What information should be available in the proof trace to be able to test it?

If instead of an algorithm that verifies proofs, we are interested in an algorithm that *finds* them, is the problem fundamentally different? Why is that?

Recall the theorems that you have proved. Most of the time, we first “find” the theorem, and “prove” it afterward. This intuition, which allows us not to carry out any enumeration (and to go beyond an enumeration) can be qualified as the “soul” of mathematics.

META-LEMMA 3.1.– Let  $\Gamma$  denote a set of wffs of  $\mathcal{S}_1$ .

If  $\vdash_{\mathcal{S}_1} A$  then  $\Gamma \vdash_{\mathcal{S}_1} A$ .

PROOF.– trivial, by application of the definition of a deduction. □

META-THEOREM 3.4 (deduction theorem).– Consider:

$\Gamma$ : set of wffs.

$A, B$ : wff of  $\mathcal{S}_1$ .

( $\Gamma, A$  means  $\Gamma \cup \{A\}$ )

$\Gamma, A \vdash_{\mathcal{S}_1} B$  iff  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B$

in particular (if  $\Gamma = \emptyset$ )

$A \vdash_{\mathcal{S}_1} B$  iff  $\vdash_{\mathcal{S}_1} A \Rightarrow B$

PROOF.– (only if):

Let  $B_1, B_2, \dots, B_n$  be a deduction starting from  $\Gamma \cup \{A\}$  ( $B_n = B$ )

Proof by induction on  $i$  that  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_i$  ( $1 \leq i \leq n$ )

1)  $i = 1$

by definition of a deduction:

i)  $B_1 \in \Gamma$

ii)  $B_1$  axiom of  $\mathcal{S}_1$

iii)  $B_1$  is  $A$  ( $B_1 \in (\Gamma \cup \{A\})$  and  $B_1 \notin \Gamma$  (case i) )

The three cases are proved as follows:

(A1)  $A \Rightarrow (B \Rightarrow A)$

$A \leftarrow B_1$

$B \leftarrow A$

$B_1 \Rightarrow (A \Rightarrow B_1)$

(ii) and MP:  $\vdash_{\mathcal{S}_1} A \Rightarrow B_1$ , hence (meta-lemma above)  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_1$

(i) and MP:  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_1$

(iii)  $\vdash A \Rightarrow A$  (example 3.9), thus  $\vdash_{\mathcal{S}_1} A \Rightarrow B_1$ , and (meta-lemma above):

$\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_1$



## 2) Induction

Suppose  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_k$   $k < i$

by definition of a deduction,

i)  $B_i$  is an axiom of  $\mathcal{S}_1$

ii)  $B_i \in \Gamma$

iii)  $B_i$  is  $A$

(ii) and (iii):  $B_i \in (\Gamma \cup \{A\})$

iv)  $B_i$  can be deduced from  $B_j, B_k$  ( $1 \leq j < i$ ) by MP, hence  $B_k$  is of the form  $B_j \Rightarrow B_i$

(i), (ii), (iii) as in case **(1)**

by the induction hypothesis:

iv)

(\*)  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow B_j$

(\*\*)  $\Gamma \vdash_{\mathcal{S}_1} A \Rightarrow (B_j \Rightarrow B_i)$

(A2)  $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

$B \leftarrow B_j ; C \leftarrow B_i$

by applying MP:

$\Gamma \vdash_{\mathcal{S}_1} (A \Rightarrow B_j) \Rightarrow (A \Rightarrow B_i)$  (\*\*)

$\Gamma \vdash_{\mathcal{S}_1} (A \Rightarrow B_i)$  (\*)

with  $i = n$  we obtain the desired proof.

(if): see exercise 3.23. □

REMARK 3.23.– This meta-theorem does not hold for all logics. □

EXERCISE 3.23.– Prove the *if* part of the deduction theorem. □

EXAMPLE 3.10.– We want to prove that  $(A \Rightarrow B), (B \Rightarrow C) \vdash_{\mathcal{S}_1} (A \Rightarrow C)$

$(A \Rightarrow B), (B \Rightarrow C), A \vdash_{\mathcal{S}_1} C$  (exercise 3.25)

by applying the deduction theorem

$(A \Rightarrow B), (B \Rightarrow C) \vdash_{\mathcal{S}_1} (A \Rightarrow C)$  □

REMARK 3.24.–

1) Only axiom schemas (A1) and (A2) were needed to prove the deduction theorem.

2) The proof technique that was used (and which is a general technique) to prove that a formal system satisfies a property goes as follows:

- prove the property for the axiom schemas;
- prove that the property is preserved by the inference rules;
- use induction on the length of the proof (deduction).

The usage of the deduction theorem in a proof is called the *method of the additional hypothesis*.

This method is extremely powerful, to convince oneself, it suffices to show that  $\vdash_{\mathcal{S}_1} A \Rightarrow A$  using this method, and compare the proof with the one given in example 3.9:  $A \vdash_{\mathcal{S}_1} A$  by definition of a deduction. We immediately obtain  $\vdash_{\mathcal{S}_1} A \Rightarrow A$  by the deduction theorem. □

### 3.4.1. Some properties of formal systems

The syntactic notions corresponding to formal systems provide too much liberty in their conception. It is therefore necessary to separate those notions that are useful from those that are not. This is the role of the following definition.

DEFINITION 3.12.– Consider a formal system  $\mathcal{S} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$ .

– An inference rule is *sound* iff the conclusion is a logical consequence of the premises.

–  $\mathcal{S}$  is *sound* iff every theorem is a valid wff.

–  $\mathcal{S}$  is *complete* (or *adequate*) iff every valid wff is a theorem.

–  $\mathcal{S}$  is *consistent* (or *coherent*) or more precisely *consistent for negation* iff there is no wff  $A \in \mathcal{L}$  such that  $\vdash_{\mathcal{S}} A$  and  $\vdash_{\mathcal{S}} \neg A$ .

–  $\mathcal{S}$  is *absolutely consistent* iff the set  $\tau \subseteq \mathcal{L}$  of the theorems of  $\mathcal{S}$  is such that  $\tau \neq \mathcal{L}$  (i.e.  $\mathcal{L}$  contains at least one wff that is not a theorem).

–  $\mathcal{S}$  is *decidable* iff there exists a mechanical procedure (algorithm) that can answer for all wffs  $A \in \mathcal{L}$  whether  $\vdash_{\mathcal{S}} A$ . Such an algorithm is called a *decision procedure*.

REMARK 3.25.– ( $\omega$ -consistency). Another notion of consistency should be mentioned here.

$\mathcal{S}$  is  $\omega$ -consistent iff for all variables  $x$  and for all formulas  $F$ , it is not the case that:

$\vdash_{\mathcal{S}} F(0), \vdash_{\mathcal{S}} F(1), \vdash_{\mathcal{S}} F(2), \dots$  and

$\vdash_{\mathcal{S}} \neg \forall x F(x)$  □

REMARK 3.26.– The notions of soundness and completeness have a natural application in computer science. Given the specification of a problem to be solved, a program meant to provide the solution(s) to the problem is *sound* if it computes correct solutions (i.e. if it computes what is specified). It is *complete* if it computes all solutions (i.e. if it computes all that is specified). □

EXERCISE 3.24.– Prove that  $\mathcal{S}_1$  is:

a) sound;

b) consistent;

c) decidable (we may assume that the completeness of  $\mathcal{S}_1$  has already been proved). □

EXERCISE 3.25.– Construct the proofs (or deductions) of the following formulas:

a)  $\vdash (\neg A \Rightarrow A) \Rightarrow A$

b)  $A \Rightarrow (B \Rightarrow C), B \vdash A \Rightarrow C$

c)  $A \Rightarrow B, B \Rightarrow C, A \vdash C$

d)  $\neg B \Rightarrow \neg A, A \vdash B$

e)  $A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C$

f)  $\vdash \neg \neg A \Rightarrow A$

g)  $\vdash A \Rightarrow \neg \neg A$

h)  $\vdash (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$

i)  $\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow (B \Rightarrow (A \Rightarrow C))$  □

REMARK 3.27.– Note that the hypotheses of (b), (c), (d), and (e) have been distinguished typographically and correspond to names that are given to particular formulas (and not to meta-variables that denote arbitrary formulas).  $A, B, C$  each denote a formula.

The reason for this is that, for example, in (b), if instead of  $B$  we had  $B$  as a hypothesis, then we could directly prove the formula using  $B \leftarrow A \Rightarrow C$ .

Nevertheless, the same deductions can also be carried out assuming that these are meta-variables, and thus the additional hypotheses and conclusions written in italics. □

EXERCISE 3.26.– Prove that in  $\mathcal{S}_1$ , consistency with respect to (w.r.t.) negation and absolute consistency coincide, i.e.  $\mathcal{S}_1$  is consistent for negation iff  $\mathcal{S}_1$  is absolutely consistent.  $\square$

### 3.4.2. Another formal system for PL (PC)

We shall name this system  $\mathcal{S}_2$ .

$\mathcal{L}$ : the language of PL using the set of connectives  $\{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$

$\mathcal{R}$ : MP

$\mathcal{A}$ : the four axiom schemas below:

$$A1) (P \vee P) \Rightarrow P$$

$$A2) Q \Rightarrow (P \vee Q)$$

$$A3) (P \vee Q) \Rightarrow (Q \vee P)$$

$$A4) (Q \Rightarrow R) \Rightarrow ((P \vee Q) \Rightarrow (P \vee R))$$

The following definitions can be used:

$$D1) P \Rightarrow Q \stackrel{\text{def}}{=} \neg P \vee Q$$

$$D2) P \wedge Q \stackrel{\text{def}}{=} \neg(\neg P \vee \neg Q)$$

$$D3) P \Leftrightarrow Q \stackrel{\text{def}}{=} (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

EXERCISE 3.27.– Give the proofs in  $\mathcal{S}_2$  of:

$$a) \vdash Q \Rightarrow (P \Rightarrow Q)$$

$$b) \vdash (P \Rightarrow \neg P) \Rightarrow \neg P$$

$$c) \vdash (P \Rightarrow \neg Q) \Rightarrow (Q \Rightarrow \neg P)$$

$$d) \vdash (Q \Rightarrow R) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$$

$$e) \vdash P \Rightarrow (P \vee P)$$

$$f) \vdash P \Rightarrow P$$

$$g) \vdash P \vee \neg P$$

$$h) \vdash P \Rightarrow \neg\neg P$$

$\square$

### 3.4.3. Another formal system

Another formal system for PL, which we shall name  $\mathcal{S}_3$  is different from  $\mathcal{S}_1$  *only* because of the axiom schemas.

The set of axiom schemas of  $\mathcal{S}_3$  (which replaces the set  $A1, A2, A3$ ) is:

- B1)  $\neg A \Rightarrow (A \Rightarrow B)$   
 B2)  $B \Rightarrow (A \Rightarrow B)$   
 B3)  $(A \Rightarrow B) \Rightarrow ((\neg A \Rightarrow B) \Rightarrow B)$   
 B4)  $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

with  $A, B, C$  denoting wffs (as in  $S_1$ ).

EXERCISE 3.28.–

a) Can the deduction (meta)theorem be used in  $S_3$ ?

b) Give a proof of:

$$\vdash_{S_3} A \Rightarrow \neg\neg A. \quad \square$$

EXERCISE 3.29.– The following questions correspond to notions that have already been manipulated. The goal is to find formal definitions of these notions, and to study what they are related to.

- a) How would you define the equivalence between two formal systems?  
 b) How would you define the independence of a set of axiom schemas?  
 c) Using the definition given in (b), give a set of axiom schemas that is not independent for PL.  
 d) How would you define the independence of a set of inference rules?  
 e) What would be the intuition behind the proof of the equivalence of two formal systems, and the independence of two sets of axioms?

Do these techniques seem to be always applicable? □

DIGRESSION 3.5.– (natural deduction systems). Formal systems, as they have been described, are known as *Hilbert systems* (or Hilbertian) or *Frege systems* (or Fregean).

There are some among the axioms and the inference rules that can be applied in all domains of mathematics, they are called *logical* axioms and inference rules, and others that depend on the domain under consideration, which are called *proper* axioms and inference rules.

Example of a logical axiom: *the law of excluded middle* (classical logic).

Example of a logical inference rule: *modus ponens*.

Example of a proper axiom: *commutativity, associativity*.

Example of a proper inference rule:

$$\frac{P(0) \text{ if } P(n) \text{ then } P(n+1)}{\forall n P(n)}$$

Another large family of formal systems is the family of *natural deduction systems*.

A natural deduction system can be viewed as a set of rules (corresponding to “natural” rules such as those that are used in informal proofs in mathematics) that determine the concept of *deduction* for a language or a set of languages. The language and the system make up a *calculus*.

Although there are different natural deduction systems, this name is considered to be a synonym of *Gentzen system* or *sequent calculus*. Sequent calculus can be viewed as a meta-calculus for the deduction relation, in the corresponding natural deduction systems.

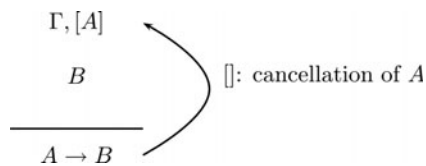
In these systems, to prove that  $B$  follows from  $A$ , or that  $A$  implies  $B$ , or that  $A \rightarrow B$ , we *assume*  $A$  and obtain  $B$  (direct proof), or we assume  $A$  and  $\neg B$ , deduce  $\neg A$ , and from this contradiction, conclude  $B$  (classical mathematics).

Natural deduction systems (or *natural deduction calculi*, we could also say *deductive systems of natural deduction*, see definition 3.9) do not contain any logical axiom, and *arbitrary* formulas can be introduced as premises.

The idea of getting rid of axioms and using conditional deductions instead originated with Gentzen and Jaškowski. Some authors believe the idea actually originated with Łukasiewicz. These systems are meant to mirror the form of intuitive reasoning and to keep the *structure* of these proofs.

It can be shown that natural deduction systems are equivalent from a deductive point of view to axiomatic formulations, i.e. if from hypotheses  $A_1, A_2, \dots, A_n$  we can derive  $C$  in a natural deduction system, then  $A_1, A_2, \dots, A_n \vdash_S C$  in an axiomatic system  $S$ , and conversely, if  $\vdash_S C$  in the axiomatic system  $S$ , then  $C$  can be derived (without any hypothesis) in a natural deduction system.

In these systems, only inference rules are given, and to prove that  $\Gamma \vdash A \rightarrow B$ , we prove that  $\Gamma, A \vdash B$  (see meta-theorem 3.4), which is frequently written in a tree-like way:



The action of putting  $[ \ ]$  around  $A$  after having written  $A \rightarrow B$  is called *cancellation* and is allowed by the *introduction* rule:

$$\frac{[A] \quad B}{A \rightarrow B}$$

of which *modus ponens* is the opposite rule, which corresponds to ( $\rightarrow$ )-*elimination*:

$$\frac{A \quad A \rightarrow B}{B}$$

We have given a foretaste of these rules in the solution to exercise 3.13.

Proofs in these systems are represented as trees, with the root at the bottom (as it is the case for real-life trees  $\smile$ ). The formula that labels the root is the logical consequence of the formulas that are not cancelled (or *open* premises), that label the leaves of the tree above the root.

The interesting thing with proofs in these systems is that the logical part (i.e. the axioms and inference rules), that can be cumbersome and uninteresting, is no longer considered. In this sense, they are closer to the usual practice of mathematics.

The following very simple example shows a *reduction* to eliminate *indirections*:

Consider the tree representing the proof of  $B$ :

$$\frac{\frac{\frac{\Pi_1 \quad [A] \quad A \quad \Pi_2 \quad B}{A \rightarrow B}}{B}}{B} \quad []: \text{cancellation of } A$$

$\Pi_1$  and  $\Pi_2$  denote derivations.

In the right branch,  $A$  was introduced and was eliminated afterward (“*detour*”).

Thus, the proof above can be reduced to (replaced by) the following.

$$\frac{\Pi_1 \quad A \quad \Pi_2 \quad B}{B} \quad \text{replace every introduction of } A \text{ in } \Pi_2$$

This digression gives a simple idea of the topics that are treated in a very important domain of logic: *proof theory*, in which proofs are the object of mathematical study. Researchers consider the normal forms of proofs, their identities, their generalizations, their complexity, etc.

This discipline was introduced by D. Hilbert, who at the beginning of the 20th Century proposed a project, the aim of which was to prove the consistency of arithmetic. For this purpose, Hilbert believed that it was necessary to study in detail formal proofs in this theory, hence the name proof theory.

EXAMPLE 3.11.– (sequent calculus: Gentzen’s LK system). These calculi use the notion of sequent<sup>15</sup> (which has already been mentioned). The notion of sequents was reused in logic programming (see section 6.2).

Capital Greek letters  $\Gamma, \Delta, \dots$  denote *finite sequences* (that may be empty) of wffs, and  $A, B, C, \dots$  will denote wffs.

$\Gamma \rightarrow \Delta$  is a sequent.

$A_1, A_2, \dots, A_m \rightarrow B_1, B_2, \dots, B_n$  ( $m, n \geq 0$ ) means

if  $A_1 \wedge A_2 \wedge \dots \wedge A_m$  then  $B_1 \vee B_2 \vee \dots \vee B_n$

$A_1, A_2, \dots, A_m \rightarrow$  means  $\neg(A_1 \wedge A_2 \wedge \dots \wedge A_m)$

$\rightarrow B_1, B_2, \dots, B_n$  means  $B_1 \vee B_2 \vee \dots \vee B_n$

### Inference rules

#### 1) Structural rules

– Weakening

$$\text{left: } \frac{\Gamma \rightarrow \Delta}{D, \Gamma \rightarrow \Delta} \qquad \text{right: } \frac{\Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, D}$$

– Contraction

$$\text{left: } \frac{D, D, \Gamma \rightarrow \Delta}{D, \Gamma \rightarrow \Delta} \qquad \text{right: } \frac{\Gamma \rightarrow \Delta, D, D}{\Gamma \rightarrow \Delta, D}$$

– Exchange

$$\text{left: } \frac{\Gamma, C, D, \Pi \rightarrow \Delta}{\Gamma, D, C, \Pi \rightarrow \Delta} \qquad \text{right: } \frac{\Gamma \rightarrow \Delta, C, D, \Lambda}{\Gamma \rightarrow \Delta, D, C, \Lambda}$$

– Cut

$$\frac{\Gamma \rightarrow \Delta, D \quad D, \Pi \rightarrow \Lambda}{\Gamma, \Pi \rightarrow \Delta, \Lambda} \quad \% \text{ corresponds to the resolution rule (see definition 3.15)}$$

<sup>15</sup> From the Latin word meaning “thing that follows”.



2) *Logical rules*

$$\neg \text{ left: } \frac{\Gamma \rightarrow \Delta, D}{\neg D, \Gamma \rightarrow \Delta}$$

$$\neg \text{ right: } \frac{D, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, \neg D}$$

$$\wedge \text{ left: } \frac{C, \Gamma \rightarrow \Delta}{C \wedge D, \Gamma \rightarrow \Delta} \text{ and } \frac{D, \Gamma \rightarrow \Delta}{C \wedge D, \Gamma \rightarrow \Delta}$$

$$\wedge \text{ right: } \frac{\Gamma \rightarrow \Delta, C \quad \Gamma \rightarrow \Delta, D}{\Gamma \rightarrow \Delta, C \wedge D}$$

$$\vee \text{ left: } \frac{C, \Gamma \rightarrow \Delta \quad D, \Gamma \rightarrow \Delta}{C \vee D, \Gamma \rightarrow \Delta}$$

$$\frac{\Gamma \rightarrow \Delta, D}{\Gamma \rightarrow \Delta, C \vee D}$$

$$\vee \text{ right: } \frac{\Gamma \rightarrow \Delta, C}{\Gamma \rightarrow \Delta, C \vee D} \text{ and}$$

A sequent of the form  $A \rightarrow A$  is called an initial sequent or axiom.

A proof  $P$  in LK is a tree (root at the bottom) of sequents such that:

- the sequents labeling the leaves are initial sequences;
- every sequent in  $P$ , except for the one at the root, is a sequent that is a premise of an inference whose conclusion is also in  $P$ .

*Proof of the law of excluded middle in LK:*

$$\begin{array}{r}
 A \rightarrow A \\
 \hline
 \rightarrow A, \neg A \qquad \neg \text{ right} \\
 \hline
 \rightarrow A, A \vee \neg A \qquad \vee \text{ right} \\
 \hline
 \rightarrow A \vee \neg A, A \qquad \text{exchange right} \\
 \hline
 \rightarrow A \vee \neg A, A \vee \neg A \qquad \vee \text{ right} \\
 \hline
 \rightarrow A \vee \neg A \qquad \text{contraction right}
 \end{array}$$

□

REMARK 3.28.— (limits: Gödel’s incompleteness theorems). Together with their elegance, formal systems are quite reassuring. As they are independent from any particular interpretation, we can imagine, thinking of their formulas and theorems, that “nothing gets past them”. This characteristic is mirrored in the *formalist* school of thought, which insisted on the *purely formal* side of mathematics (i.e. an activity entirely controlled by the rules of the game), and of which the great mathematician D. Hilbert was one of the principal advocates.

In what is called “Hilbert’s program”, Hilbert, who wanted to obtain sound foundations for mathematics, stated the problem of finding a formal system (including arithmetic and mathematical analysis) capable of discovering all mathematical truths and only those, and not containing any contradiction (i.e. a *consistent* formal system). Hilbert wanted to prove the consistency of mathematics using mathematical *finitary*<sup>16</sup> methods. Such a result would talk about proofs, it would be a result of *meta-mathematics* (sometimes we talk about *proof theory* or *meta-mathematics*).

The hopes of Hilbert’s program were crushed by both of *Gödel’s incompleteness theorems*. K. Gödel (1906–1978), who is considered as one of the greatest logicians of all times, showed the distinction between what is *true* and what is *provable*.

In his *first incompleteness theorem* (see section 5.9), he showed that given a theory  $\mathcal{T}$  containing arithmetic and assumed to be consistent, in which there are either finitely many axioms and inference rules or they can be specified recursively, one can construct formulas  $G$  (in the language of  $\mathcal{T}$ ) that are undecidable, i.e.  $\not\vdash_{\mathcal{T}} G$  and  $\not\vdash_{\mathcal{T}} \neg G$  (sometimes this theorem is stated by saying that there exist in  $\mathcal{T}$  true formulas that are unprovable).

The formula  $G$  is, for example, “I am not provable”. If  $\vdash_{\mathcal{T}} G$  and  $\mathcal{T}$  is consistent then  $G$  would be true, but  $G$  precisely states that it is unprovable. Contradiction. Hence,  $G$  is unprovable.

Arithmetic enables us to encode formulas and proofs as numbers and to state their properties as properties of integers.

In the system  $\mathcal{T}$ , it is possible to encode a formula  $Con_{\mathcal{T}}$  whose interpretation is “ $\mathcal{T}$  is consistent”. Gödel’s *second incompleteness theorem* states that  $\not\vdash_{\mathcal{T}} Con_{\mathcal{T}}$ , which means that it is impossible to prove the consistency of  $\mathcal{T}$  in  $\mathcal{T}$ .  $\square$

### 3.5. The method of Davis and Putnam

The importance of this method and that of the SAT problem are closely related.

---

<sup>16</sup> Although Hilbert did not specify what he meant by “finitist”, all finitary methods can probably be formulated in the ZFC formalization (ZF + AC) in set theory.

This method is applied to sets of clauses  $S$ , or, equivalently, to cnf formulas (see definition 3.8 for the terminology).

It permits us to decide whether  $S$  is satisfiable or not, and if it is satisfiable, to produce models of  $S$ .

The underlying idea is very simple. If we want to detect whether a set of clauses admits models, then we consider the literals one by one. A literal  $L$  can be evaluated either to **T** or to **F**. If  $L$  is evaluated to **T**, then any clause containing  $L$  can be ignored, and if  $L$  is evaluated to **F**, then  $L$  can be erased from any clause it occurs in. Of course, the same rules apply to  $L^c$ .

The method uses the following rules:

**R-0** Remove all clauses that contain tautologies (i.e. clauses of the form  $L \vee \neg L \vee \alpha$ ).

**R-1** a) If  $S$  contains two complementary unit clauses, then  $S$  is unsatisfiable.

b) (*unit clause rule*) If (a) does not apply and if  $S$  contains a unit clause  $L$ , then remove all clauses containing  $L$  and all occurrences of  $L^c$  from the other clauses.

**R-2** (*pure literal rule*) If  $L$  occurs in  $S$ , but  $L^c$  does not, all clauses containing  $L$  can be removed.

**R-3** (*splitting rule*) If  $S$  contains non-unit clauses in which  $L$  and  $L^c$  occur, then replace  $S$  by  $S_1$  and  $S_2$ .

$S_1$  is the set of clauses in which all occurrences of  $L$  have been removed, as well as all the clauses containing  $L^c$ .

$S_2$  is the set of clauses in which all occurrences of  $L^c$  have been removed, as well as all the clauses containing  $L$ .

**R-4** (*subsumption rule*) If  $S$  contains a clause of the form  $L \vee \alpha$ , remove all clauses of the form  $L \vee \alpha \vee \beta$  ( $\alpha$  and  $\beta$  are disjunctions of literals).

REMARK 3.29.— We shall also apply R-2 and R-4 when considering the resolution rule (see section 3.7).  $\square$

The algorithm DP applies rules **R-1** to **R-4** and enables us to detect the satisfiability (or unsatisfiability) of a set of clauses of PL. It is simple to verify that rule **R-0** can be applied at a preprocessing phase, as the other rules cannot generate tautologies (they *divide* or *eliminate* clauses).

EXAMPLE 3.12.— Consider the set of clauses

$$S = \{P \vee Q, \neg Q \vee S, \neg S \vee P, \neg P \vee R, \neg R \vee \neg P \vee T, \neg T \vee \neg R\}$$

```

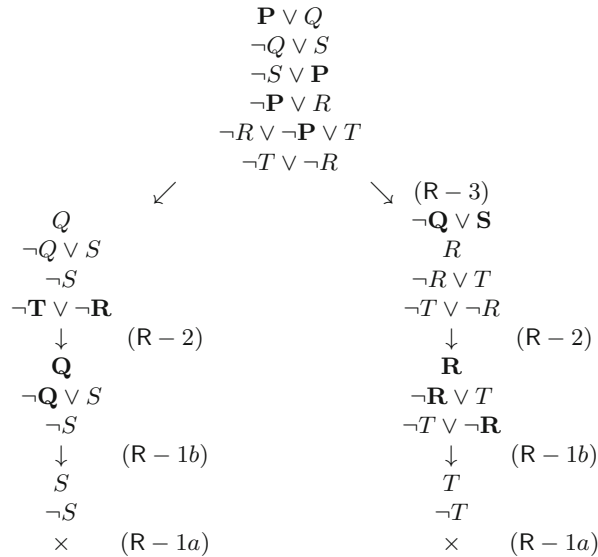
algorithm DP ( $S$ )
input: a finite set of clauses of PL:  $S$ 
output: unsat or sat
begin
  case
  • if  $S = \emptyset$  return sat
  • if R-1 (a) applies (i.e.  $L \in S$  and  $L^c \in S$ ) return unsat
  • if R-2 applies and yields  $S_1$  then
    if DP ( $S_1$ ) = sat then return sat else return unsat
  • if R-1 (b) applies and yields  $S_1$  then
    if DP ( $S_1$ ) = sat then return sat else return unsat
  • if R-3 applies and yields  $S_1$  and  $S_2$  then
    if DP ( $S_1$ ) = sat or DP ( $S_2$ ) = sat then return sat else
    return unsat
  end-case
end
  
```

Figure 3.2. Davis and Putnam algorithm (DP)

that corresponds to the following cnf wff:

$$(P \vee Q) \wedge (\neg Q \vee S) \wedge (\neg S \vee P) \wedge (\neg P \vee R) \wedge (\neg R \vee \neg P \vee T) \wedge (\neg T \vee \neg R)$$

that shall be represented as a matrix. The deduction proving that  $S$  is contradictory is as follows:



□

EXERCISE 3.30.– We consider the following deductive system (i.e. with no logical axioms:  $\mathcal{A} = \emptyset$ , see definition 3.9):

$$\mathcal{S}_{DP} = \langle \mathcal{L}, \mathcal{R} \rangle$$

where:

$\mathcal{L}$ : sets of clauses.

$$\mathcal{R} = \{ \text{R-0, R-1, R-2, R-3, R-4} \}$$

Prove that the Davis–Putnam method is sound and complete (see definition 3.12).

Here we can translate:

*sound*: **if**  $S \vdash_{\mathcal{S}_{DP}} (S \text{ unsat})$  **then**  $S \text{ unsat}$ ;

% which means “the method can be trusted”

*complete*: **if**  $S \text{ unsat}$  **then**  $S \vdash_{\mathcal{S}_{DP}} (S \text{ unsat})$

% which means “we can detect *all* unsatisfiable sets of clauses with this method”. □

### 3.5.1. The Davis–Putnam method and the SAT problem

In the literature, this method is often presented as a means to solve the SAT problem.

Inspiring from the soundness and completeness proofs (see solution to exercise 3.30), it is simple to obtain the algorithm that constructs the models of satisfiable sets of clauses.

Example 3.13 clearly shows the stages of the algorithm for model construction.

The two following properties, whose justification is trivial, are very useful to design the algorithm (the first one is not used in the example).

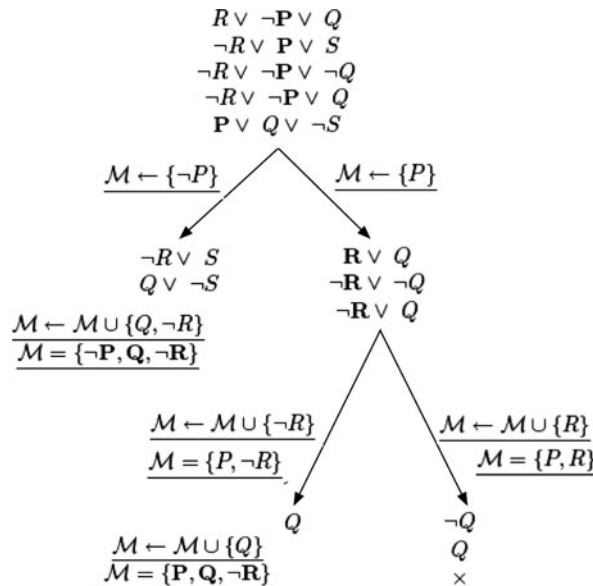
Let  $S$  denote the set of clauses under consideration,  $\mathcal{M}$  is a set specifying the potential model that is currently being built.

– If  $C \in S$  and  $C$  is a unit clause (i.e.  $C$  contains only one literal  $L$ ), then necessarily  $L \in \mathcal{M}$ .

–  $L \in C \in S$  and  $L$  is pure then  $\mathcal{M} \leftarrow \mathcal{M} \cup \{L\}$  (at the start  $\mathcal{M} \leftarrow \{L\}$ ) is a model of  $S$ .

REMARK 3.30.– If a clause becomes a unit clause and/or a literal becomes pure by application of the rules and  $L^c \in \mathcal{M}$ , then the model is not viable.  $\square$

EXAMPLE 3.13.– Determine whether the set of clauses  $\mathcal{E}$  below is satisfiable or not. If it is, give models of this set.



We have therefore constructed four models for  $\mathcal{E}$ :

$$\{\neg P, Q, \neg R, S\}, \{\neg P, Q, \neg R, \neg S\}, \{P, Q, \neg R, S\}, \{P, Q, \neg R, \neg S\}.$$

Of course, we could have stopped searching after obtaining the first model (for example  $\{\neg P, Q, \neg R, S\}$ ).  $\square$

EXERCISE 3.31.– For the set of clauses  $S$  of example 3.13, is it possible to find other models by applying the method of Davis and Putnam with another strategy?  $\square$

### 3.6. Semantic trees in PL

We start by noticing that semantic trees method  $\neq$  semantic tableaux methods.

– The method of semantic tableaux is used to enumerate *models* (partial models in first-order logic) of a *set of wffs*.

– The method of semantic trees is used to enumerate the *interpretations* (partial interpretations in first-order logic) of a *set of clauses*.

DEFINITION 3.13.– Let  $S$  be a set of clauses, the base of  $S$ , denoted by  $B(S)$  is defined as follows:

$$B(S) = \{L \mid L \text{ positive and } [(L \in C \in S) \text{ or } (L^c \in C \in S)]\}$$

Given an enumeration of the elements in  $B(S)$ :

$$B(S) = \{L_1, L_2, \dots, L_n\}$$

(or  $B(S) = \{L_1, L_2, \dots, L_n, \dots\}$  if  $S$  is infinite).

– A semantic tree for  $S$  is a binary tree with branches that are labelled as follows:

$$f_l(n_i^j) = L_{i+1} \quad f_r(n_i^j) = \neg L_{i+1}$$

$$(L_i \in B(S); \quad 0 \leq i \leq n-1)$$

$$f_l: \text{ left son} \quad f_r: \text{ right son}$$

( $i \geq 0$ ): depth of the node; ( $1 \leq j \leq 2^{i+1}$ ): position from the left-hand side to the right-hand side.

It is clear from the definition that:

– a branch cannot contain  $L \in C \in S$  and  $L^c \in D \in S$ ;

– the set of branches corresponds to the set of interpretations of  $S$ .

– The node  $n_i$  (for the least value of  $i$ ) whose branch (interpretation) going through  $n_i$  is a counter model of a clause in  $S$  is called a failure node (denoted by  $\times$ ). A branch containing a failure node is a closed branch. A branch that is not closed is open and corresponds to a model of  $S$  (when  $S$  is infinite, an open branch is necessarily infinite).  $\square$

– A semantic tree is closed iff all its branches are closed (i.e. all its leaves are failure nodes). Otherwise, it is open.

– A node is an inference node iff its immediate descendants are failure nodes.

THEOREM 3.2.–  $S$ : finite set of clauses.

$S$  is unsatisfied iff there exists a closed semantic tree  $T$  for  $S$ .

PROOF.– If,

Every closed branch is a counter model of a clause in  $S$ , and therefore, of  $S$ . As the semantic tree enumerates *all* the interpretations and  $T$  is closed,  $S$  must be unsatisfiable.

Only if,

$S$  is unsatisfiable, hence no interpretation can be a model of  $S$ ; thus, there cannot be any open branch  $B_0$  in  $T$ . Otherwise,  $B_0$  would not falsify any clause in  $S$  and would therefore be a model of  $S$ : contradiction.  $T$  is necessarily closed.  $\square$

EXAMPLE 3.14.– Consider the set of clauses:

$$S = \{C_1, C_2, C_3, C_4\}$$

where:

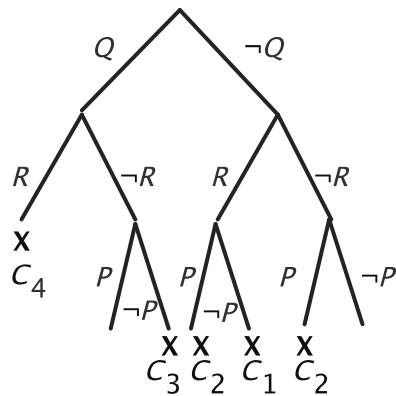
$$C_1: P \vee Q \vee \neg R$$

$$C_2: \neg P \vee Q$$

$$C_3: P \vee \neg Q \vee R$$

$$C_4: \neg Q \vee \neg R$$

$$B(S) = \{P, Q, R\}$$



We have, therefore, found two models among the eight possible interpretations:

$$\{Q, \neg R, P\} \text{ and } \{\neg Q, \neg R, \neg P\}$$

and six counter models:

$$\{Q, R, P\}, \{Q, R, \neg P\}, \{Q, \neg R, \neg P\}, \{\neg Q, R, P\},$$

$$\{\neg Q, R, \neg P\}, \text{ and } \{\neg Q, \neg R, P\}.$$

$\square$



EXERCISE 3.32.–

a) Give a semantic tree (there are many of them, depending of the order chosen on  $B(S)$ ) for the set of clauses:

$$S = \{C_1, C_2, C_3, C_4, C_5, C_6\}$$

with:

$$C_1: P \vee Q$$

$$C_2: \neg Q \vee S$$

$$C_3: P \vee \neg S$$

$$C_4: \neg P \vee R$$

$$C_5: \neg P \vee \neg R \vee T$$

$$C_6: \neg R \vee \neg T$$

b) Mark out all the inference nodes. What meaning do these nodes have?

c) Give a semantic *tableaux* for  $S$ . □

EXERCISE 3.33.– Give a semantic tree for the set of clauses below:

$$C_1: \neg P \vee \neg Q \vee R$$

$$C_2: P \vee R$$

$$C_3: Q \vee R$$

$$C_4: \neg R$$
□

REMARK 3.31.– In the proof of the following theorem, we shall extend the definition of semantic trees to *arbitrary* wffs of PL without loss of generality.

The definition of a tree is the same. The only difference is that there is no uniform way to close a branch, as it was the case for sets of clauses, but we must take into account the connectives that occur in the formula that is evaluated in the partial interpretation under consideration.

Another possibility is to consider an *equivalent* set of clauses for each wff. □

THEOREM 3.3 (compactness of PL).–  $S$  a set of wffs of PL.

*If every finite subset of  $S$  is satisfiable then  $S$  is satisfiable.*

PROOF.–

– If  $S$  is finite then the proof is trivial:  $S$  is a finite subset of  $S$  and is satisfiable by hypothesis.

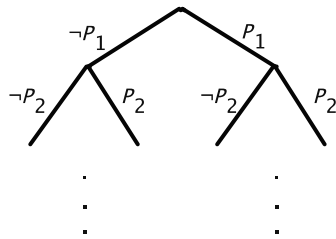
– If  $S$  is infinite,

i)  $B(S) < \infty$ , necessarily, there are infinitely many formulas that contain the *same* propositional symbols, they only differ by the number of occurrences of these symbols and, of course, by the number of connectives.

The corresponding semantic tree is therefore necessarily finite, and it is either closed (same reasoning as in (a) below), or it is open, and any open branch is a model of  $S$ .

ii)  $B(S) = \{P_1, P_2, \dots, P_n, \dots\}$

A semantic tree for this case is:



Warning: this figure does not entail that the set of interpretations for a denumerably infinite number of base symbols is denumerably infinite (see exercise 3.1).

There are two cases to consider.

a) All the branches are closed (at a *finite* depth); in this case (same reasoning as for finite semantic trees)  $S$  is unsatisfiable.

If  $n \in \mathbb{N}$  is the maximal depth of the failure nodes, then the *finite subset* of  $S$

$S' : \{F_i \mid \bigcup_i \text{Propset}(F_i) = \{P_1, P_2, \dots, P_n\}\}$  with  $\text{card}(S') \leq 2^n$  (i.e.  $S'$  is finite) is *unsatisfiable*.

$\text{Propset}(F_i)$  denotes (as usual) the set of base symbols in  $F_i$ .

We have proved that if  $S$  is *unsatisfiable*, then there exists a *finite subset* of  $S$  that is also unsatisfiable, i.e. the contrapositive.

b) There exists at least an open branch. It is necessarily infinite (there are infinitely many formulas) and it represents a model of  $S$  (as it does not falsify any formula in  $S$ ). □

REMARK 3.32.– This theorem is essential for the definition of a semi-decision procedure for first-order logic (see theorem 5.7).

To get an idea, consider the following expression:

1) for all  $x \in \mathbb{N}$ ,  $x \geq 0$

and the *infinite* set of propositions:

2)  $\{0 \geq 0, 1 \geq 0, 2 \geq 0, \dots\}$

(1) and (2) have the same meaning.

The compactness theorem does not apply for all logics. To convince, it suffices, for example, to consider the following set:

$S: \{x \in \mathbb{N}, x \neq 1, x \neq 2, x \neq 3, \dots\}$

Every finite subset of  $S$  is satisfiable, but  $S$  is *unsatisfiable*.  $\square$

### 3.7. The resolution method in PL

This method is one of the most widely used in practice (especially its version for first-order logic). It uses a unique inference rule, which makes it particularly easy to implement, but it needs a normal form: the *clausal form* (or cnf) (see definition 3.8)<sup>17</sup>. We begin by a remark.

REMARK 3.33.–

– The method requires a set of clauses (cnf) as an input. This is not a limitation since any wff in PL can be transformed into an equivalent formula in cnf.

– A clause (or a set containing a clause) is, by definition, satisfiable.

–  $P \wedge \neg P$  is not a clause, although this contradiction is represented by the so-called empty clause (denoted by  $\square$ ).  $P$  is a unit clause and  $\neg P$  is another one.

– We indifferently consider a wff in clausal form as a wff in cnf or as a *set* of clauses, and clauses as *sets* of literals.  $\square$

DEFINITION 3.14.– Let  $S = \{C_1, \dots, C_n\}$  denote a set of clauses. A set of literals  $M$  is a model of  $S$  iff:

if  $L \in M$  then  $L^c \notin M$  and:

$$C_k \cap M \neq \emptyset \quad (1 \leq k \leq n)$$

---

<sup>17</sup> There also exists a *non-clausal* resolution.

(This definition can be expressed in English by saying: to evaluate a set of clauses to  $\mathbf{T}$ , all its clauses must be evaluated to  $\mathbf{T}$ . A literal cannot be evaluated to  $\mathbf{T}$  and  $\mathbf{F}$  simultaneously. To evaluate a clause to  $\mathbf{T}$ , at least one of its literals must be evaluated to  $\mathbf{T}$ ).

As a consequence, if all the literals in a clause are evaluated to  $\mathbf{F}$ , then the clause will be evaluated to  $\mathbf{F}$ .

DEFINITION 3.15.– Given two clauses containing complementary literals:

$C_1: L \vee \alpha$  ( $\alpha$ : disjunction of literals, i.e. a clause (see definition 3.8);

$C_2: L^c \vee \beta$  ( $\beta$ : disjunction of literals, i.e. a clause (see definition 3.8).

The inference rule, named the resolution rule is defined as follows:

$$R: \frac{L \vee \alpha \quad L^c \vee \beta}{\alpha \vee \beta}$$

we will also note it (see remark 3.33):

$R(C_1, C_2) = C$  or, if in order to underline the complementary literals,

$R(C_1, C_2, L, L^c) = C$

where:

$$C = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{L^c\})$$

The clause  $C: \alpha \vee \beta$  is called the resolvent of  $C_1$  and  $C_2$ .

$C_1$  and  $C_2$  are the parent clauses

$C$  is a logical consequence of  $\{C_1, C_2\}$  (of  $C_1 \wedge C_2$ ), but  $C$  is not equivalent to  $C_1 \wedge C_2$  (every model of  $\alpha$  (respectively,  $\beta$ ) is a model of the resolvent, but not necessarily of both parent clauses).

In the case in which  $\alpha$  and  $\beta$  do not contain any literal:

$$\frac{L \quad \neg L}{\square}$$

where  $\square$ , which denotes a contradiction, is called the empty clause.

We shall use the rule:

$$\text{Abs} : \frac{\alpha \vee L \vee \beta \vee L \vee \beta \vee \gamma}{\alpha \vee L \vee \beta \vee \gamma}$$

(where  $\alpha$ ,  $\beta$ , and  $\gamma$  are disjunctions of literals). This rule boils down to considering clauses as sets of literals (see definition 3.8 or equivalently) to using the associativity, commutativity, and absorbing properties of  $\vee$  (i.e.  $(\alpha \vee \beta) \vee \gamma \Leftrightarrow \alpha \vee (\beta \vee \gamma)$ ,  $L \vee \alpha \Leftrightarrow \alpha \vee L$  and  $L \vee L \Leftrightarrow L$ , respectively).

REMARK 3.34.– (empty clause  $\neq$  empty set of clauses). It is very important not to confuse the empty clause with an empty set of clauses. The former is unsatisfiable and the latter is satisfiable (the set does not contain anything, it cannot contain a contradiction).

We can provide a more formal proof. By *reductio ad absurdum*: if  $\emptyset$  is unsatisfiable, then (for example)  $\{A \vee B\} = \emptyset \cup \{A \vee B\}$  would be unsatisfiable, as it contains an unsatisfiable subset. However,  $\{A \vee B\}$  is satisfiable (models  $\{A\}$ ,  $\{B\}$ , and  $\{A, B\}$ ). A contradiction, so  $\emptyset$  is satisfiable.  $\square$

For non-deterministic rules such as resolution, it is useful to define an operator that permits us to capture all resolvents that can be obtained by applying the resolution rule in every possible way.

DEFINITION 3.16.– ( $\mathcal{R}$  operator). Let  $S$  denote a (finite) set of clauses:

$$\mathcal{R}(S) = S \cup \{R(C_1, C_2) \mid C_1, C_2 \in S\}$$

$$\mathcal{R}^0(S) = S$$

$$\mathcal{R}^{n+1}(S) = \mathcal{R}(\mathcal{R}^n(S)) \text{ pour } n \geq 0$$

$$\mathcal{R}^*(S) = \bigcup_{n \geq 0} \mathcal{R}^n(S)$$

REMARK 3.35.– It is clear that for a finite set of clauses  $S$  that is satisfiable, there exists an  $n$  such that:

$$\mathcal{R}^*(S) = \mathcal{R}^n(S)$$

(see exercise 3.36).  $\square$

REMARK 3.36.– (dual of resolution). The dual<sup>18</sup> of the resolution rule, the consensus rule, existed before and applies to the test of the *validity* of dnf formulas, and to their simplification. The disjuncts are also called *clauses*.

<sup>18</sup> The dual of the expression  $\bigvee(x, y, \dots, z)$  is defined as  $\neg(\bigwedge(\neg x, \neg y, \dots, \neg z))$ , where  $x, y, \dots, z$  are literals.

The consensus rule applies to (conjunctive) clauses assumed to be non-contradictory, non-tautological, and is defined as follows:

$$Cons : \frac{L \wedge \alpha \quad L^c \wedge \beta}{\alpha \wedge \beta}$$

$\alpha, \beta$ : conjunctive clauses.

The conjunctive clause  $\alpha \wedge \beta$  is called the *consensus* of  $L \wedge \alpha \vee L^c \wedge \beta$ .

It is simple to check that every model of  $\alpha \wedge \beta$  is a model of  $L \wedge \alpha \vee L^c \wedge \beta$ . Here, the dual of the empty clause denotes  $L \vee L^c$ , and is therefore a tautology. As the disjunction of two clauses is a logical consequence of their consensus and as the logical consequence of a tautology can only be a tautology, obtaining the dual of the empty clause proves the validity of the initial formula.  $\square$

DEFINITION 3.17.– (a deductive system for resolution).

$$\mathcal{S}_R = \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$$

$\mathcal{L}$ : clauses and sets of clauses

$$\mathcal{R} = \{R, Abs\} \quad \% R, Abs \text{ from definition 3.15}$$

$$\mathcal{A} = \emptyset$$

Clause  $C$  is deduced by resolution from the set of clauses  $S$ , denoted by:

$$S \vdash_{\mathcal{S}_R} C \text{ (or } S \vdash_{\mathcal{R}} C)$$

iff:

there exists a finite sequence  $C_1, \dots, C_k$

and:

$$C_k = C$$

$$C_{i+1} = R(C_m, C_n) \quad (0 \leq i \leq k-1)$$

$$C_m, C_n \in S \cup \{C_1, \dots, C_i\}$$

the sequence  $C_1, \dots, C_k$  is called a deduction starting from  $S$ , and if  $C = \square$ , it is called a refutation of  $S$ .

For the resolution method, soundness and completeness (for refutation; see also definition 3.12) are stated as follows:

soundness:  $S \vdash_{\mathcal{R}} \square$  then  $S$  is unsatisfiable (contradictory);

completeness (for refutation, or refutational completeness): if  $S$  is unsatisfiable (contradictory) then  $S \vdash_{\mathcal{R}} \square$

REMARK 3.37.– The expression *completeness for refutation* applied to the resolution method can easily be explained by noticing, for example, that:

$$A \not\vdash_{S_R} A \vee B$$

Although  $A \vee B$  is obviously a logical consequence of  $A$ .

However, by negating  $A \vee B$ , we obtain a set of clauses  $\{A, \neg A, B\}$  from which the clause  $\square$  is immediately obtained using the resolution rule between  $A$  and  $\neg A$ .  $\square$

THEOREM 3.4.– Let  $S$  denote a satisfiable set of clauses and  $M$  a model of  $S$ ,

$$\text{If } S \vdash_{S_R} C, \text{ then } M \cap C \neq \emptyset$$

PROOF.–

$$C_1 \in S \quad C_2 \in S \quad L \in C_1 \quad L^c \in C_2$$

$$R(C_1, C_2, L, L^c) = C = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{L^c\})$$

As  $M$  is a model of  $S$ ,  $M$  is a model of all the clauses in  $S$ , hence:

$$M \cap C_1 \neq \emptyset \text{ and } M \cap C_2 \neq \emptyset.$$

There are three cases to consider:

i) if  $L \in M$  and as  $M$  is a model of  $C_1$  and  $C_2$ , there exists  $K \in C_2 \setminus \{L^c\}$  and  $K \in M$ ;

hence, by definition of rule  $R$ :  $K \in C$ , thus:

$$M \cap C \neq \emptyset;$$

ii) if  $L^c \in M$ , then there exists  $N \in C_1 \setminus \{L\}$  and  $N \in M$   
 $N \in C$  (by definition of rule  $R$ ) thus:

$$M \cap C \neq \emptyset;$$

iii) If  $L \notin M$  and  $L^c \notin M$ , then  $M$  does not depend on the values assigned to  $L$  and  $L^c$ , hence:

$$M \cap C \neq \emptyset.$$

The proof is completed by applying the definition of a deduction and by induction on the length of the deduction.  $\square$

REMARK 3.38.– This theorem can be stated with another notation:

$$\text{If } \models_{\mathcal{I}} S, \text{ then } \models_{\mathcal{I}} C \quad \%(C \in \mathcal{R}(S))$$

We have used the contrapositive, i.e.

$$\text{if } \not\models_{\mathcal{I}} C \text{ then } \not\models_{\mathcal{I}} S$$

to close branches in the semantic tree, and it is also used in very efficient SAT solvers (programs that solve the SAT problem) to prune the search space. Indeed, when we verify that the proposed interpretation (partial in general, but that can be sufficient to evaluate some clauses) falsifies a clause (original or deduced by resolution), there is no need to keep going in the same direction.  $\square$

COROLLARY 3.1.– (soundness of resolution). *The resolution method is sound.*

PROOF.– Trivial, using the previous theorem.

$$\text{If } S \vdash_{S_R} \square$$

and  $S$  is satisfiable, then  $\square$  would be satisfiable, which is impossible

Hence:

$$S \text{ is unsatisfiable.} \quad \square$$

EXERCISE 3.34.– Prove the refutational completeness of the resolution method.  $\square$

EXERCISE 3.35.– Prove that tautologies can be eliminated from a refutation by resolution, without losing refutational completeness.  $\square$

EXAMPLE 3.15.– This example exhibits many features. We want to use the resolution method to prove that the set of clauses:

$$S = \{\neg P \vee \neg Q \vee R, P \vee R, Q \vee R, \neg R\}$$

is unsatisfiable, in order to design a *program* later on that will do the same thing.



The main problem is how to handle *non-determinism* (i.e. the *choices* when the resolution rule is applied). Before analyzing the good choices for the application of the rule and to be sure that the method will work in all cases, we decide to apply the “brute force method”, i.e. we apply all choices for a given enumeration and we check whether we obtain (in the original set of clauses along with those that are deduced) two complementary unit clauses (the only contradiction that can always be detected mechanically).

The notation on the right-hand side of the formulas:

$$(i, j) - (k, l) \quad (1 \leq i \leq 4; \quad 1 \leq j \leq 3; \quad 2 \leq k \leq 12; \quad 1 \leq l \leq 2)$$

means that we apply the resolution rule by choosing the literal at position  $j$  (from left to right) of clause number  $i$ , and its complement at position  $l$  in clause number  $k$ .

This notation will show its utility for resolution in first-order logic.

- |     |                                   |                    |
|-----|-----------------------------------|--------------------|
| 1)  | $\neg P \vee \neg Q \vee R$       |                    |
| 2)  | $P \vee R$                        |                    |
| 3)  | $Q \vee R$                        |                    |
| 4)  | $\neg R$                          |                    |
| 5)  | $\neg Q \vee R$                   | $(1, 1) - (2, 1)$  |
| 6)  | $\neg \mathbf{P} \vee \mathbf{R}$ | $(1, 2) - (3, 1)$  |
| 7)  | $\neg P \vee \neg Q$              | $(1, 3) - (4, 1)$  |
| 8)  | $P$                               | $(2, 2) - (4, 1)$  |
| 9)  | $Q$                               | $(3, 2) - (4, 1)$  |
| 10) | $\neg Q \vee R$                   | $(1, 1) - (8, 1)$  |
| 11) | $\neg P \vee R$                   | $(1, 2) - (9, 1)$  |
| 12) | $\mathbf{R}$                      | $(2, 1) - (6, 1)$  |
| 13) | $\square$                         | $(4, 1) - (12, 1)$ |

Note that a same clause can be deduced more than once (for example, 5 and 10; 6 and 11).

Compare this to the closed tree (that corresponds to the same set of clauses) of exercise 3.33, in which we “stumbled upon” the correct construction order for the tree.

After an analysis of the refutation once it is obtained, it turns out that only 6 and 12 were necessary to detect a contradiction. Does it seem possible to know this *before* the refutation?  $\square$

EXERCISE 3.36.– How can we detect that a set of clauses is satisfiable using the resolution rule? Give an example.  $\square$

EXERCISE 3.37.– Use the resolution method to prove the following results:

a) Prove that  $\mathcal{S}$  is unsatisfiable:

$$\mathcal{S} = \{P, \neg P \vee Q, \neg Q \vee R, \neg Q \vee \neg R\}$$

b) Prove that  $\mathcal{S}$  is unsatisfiable:

$$\mathcal{S} = \{R, Q \vee \neg R, S \vee \neg R, P \vee \neg Q \vee \neg S, \neg P \vee \neg Q \vee \neg S\}$$

c) Is  $\mathcal{S}$  satisfiable or unsatisfiable?

$$\mathcal{S} = \{P \vee Q, P \vee \neg Q, R \vee Q, R \vee \neg Q\}$$

d) Prove that  $\mathcal{S}$  is unsatisfiable:

$$\mathcal{S} = \{\neg P, \neg R \Rightarrow W, Q \vee (\neg T \Rightarrow \neg P \wedge \neg S), \neg P \Rightarrow (S \wedge \neg R), \neg Q, \neg S, \neg T, \neg R \Rightarrow Y\}$$

e) Prove, first by using any method you would have chosen when you did not know the resolution rule, then by resolution, that the following reasoning is correct:

$$\begin{array}{l} A \wedge B \Rightarrow C \wedge D \\ E \wedge F \Rightarrow G \\ G \wedge D \Rightarrow H \\ A \\ B \\ F \\ E \\ \hline H \end{array}$$

f) Prove, using the resolution rule, that the reasoning of exercise 3.8 is not correct (using the formalization that is given with its solution).  $\square$

REMARK 3.39.– The following remarks are direct consequences of the definitions:

– every subset of a *satisfiable* set of clauses (and more generally of a set of wffs) is *satisfiable*;

– every superset of an *unsatisfiable* set of clauses (and more generally of a set of wffs) is *unsatisfiable*.  $\square$

DEFINITION 3.18.– An *unsatisfiable set of clauses*  $S$  is *minimally unsatisfiable* iff for all  $R \subset S$  (i.e. for all  $R \subseteq S$ ,  $R \neq S$ ),  $R$  is *satisfiable*.

EXERCISE 3.38.– A minimally unsatisfiable set of clauses does not contain any pure literal (see exercise 3.30).

Is an unsatisfiable set of clauses that does not contain any pure literal necessarily minimally unsatisfiable?  $\square$

**THEOREM 3.5.**— *If  $S$  is unsatisfiable then there does not exist any interpretation that falsifies all the clauses in  $S$ .*

**PROOF.**— If  $S$  contains clauses with pure literals, then they can be eliminated (see exercise 3.30).

Assume that there exists an interpretation  $\mathcal{I}$  that falsifies *all* the clauses in  $S$ . By definition of a clause,  $\mathcal{I}$  evaluates *all* the literals in  $C$  to **F**.

As all clauses with pure literals have been removed from  $S$ , if  $L \in C \in S$ , then there exists  $L^c \in D \in S$ .  $L^c$  is evaluated to **T**, so that (by definition of a clause)  $D$  is also evaluated to **T**. A contradiction. Therefore,  $\mathcal{I}$  cannot exist.  $\square$

**COROLLARY 3.2.**— *If  $S$  is an unsatisfiable set of clauses and  $\mathcal{I}$  is an interpretation for  $S$ , then there exists  $S_1 \subset S$ ,  $S_2 \subset S$ ,  $S_1 \neq \emptyset$ ,  $S_2 \neq \emptyset$ ,  $S_1 \cap S_2 = \emptyset$  such that  $\mathcal{I}$  is a model of  $S_1$  and a counter model of  $S_2$ .*  $\square$

**REMARK 3.40.**— We have seen different proof procedures also called *calculi* (formal systems or “à la Hilbert”, tableaux, resolution, etc.). We may then wonder “does there exist a proof procedure that is uniformly (i.e. for all problems) better (for example, in the number of steps) than the others?”. The answer (as we might expect) is no.  $\square$

A notion that is naturally associated to the non-determinism problem is the notion of *strategy*. This word has a technical meaning that is very similar to the one of everyday language.

A strategy is a rule that is used to select the application choices of an (several) inference rule(s) to reach a certain goal, in general, by reducing the number of choices, hence the search space (i.e. the set of all possible applications before finding the desired result or stopping). Sometimes, the goal is to reduce the number of steps before reaching a solution.

### 3.8. Problems, strategies, and statements

A very large class of problems can be defined in an abstract way as a triplet  $(E, I, G)$ , where  $E$  is an environment,  $I$  an initial state, and  $G$  a goal to reach. The search of the proof of a theorem is a particular instance of this triplet (with  $E$ : a theory,  $I$ : the hypotheses, and  $G$ : the formula to prove). We shall come back to this topic later.

The resolution of problems generally requires non-determinism to be handled. Finding a “good way” of handling non-determinism is of the utmost importance. The study of how non-determinism can be handled concerns *planning* and *strategies*,

and is part of AI (in particular automated deduction and proof assistants), of operations research, of complexity theory, of robotics, etc.

These topics have been widely studied, but another problem that is as important, although it is less studied, is the *statement* of the problem. Here, it is necessary to distinguish between the statement of a problem in *different* languages or logics<sup>19</sup> from modifications of the statement in the *same logic* (see Chapter 9).

### 3.8.1. Strategies

From the start, people realized that it would be impossible to deal with interesting problems of mechanical proofs without associating strategies to the calculi (calculi are sets of non-deterministic inference rules).

Of course, people wondered what the best way of handling non-determinism would be, via a perfect procedure (strategy), i.e. that never generates any redundant formula, no matter the problem to be solved.

The non-existence of such a procedure is intuitively obvious (knowing exactly what information is required to prove a theorem boils down to knowing how to prove this theorem).

It can be shown (using well-known results from computability theory) that there are no complete procedures for refutation (for example, resolution) that are perfect, i.e. that never generate formulas (clauses) that are not necessary for the proof (refutation).

We can define a *proof procedure* in an abstract way, as a couple  $(T, \Sigma)$ , where  $T$  is a formal system (see definition 3.9) and  $\Sigma$  is a strategy for  $T$ .

It is interesting to note that, in general, books on logic only mention *proof systems* by identifying them with  $T$  without any mention to the strategy.

To define the abstract notion of an automated proof, we define the notion of a *proof graph*, which naturally follows the definition of operator  $\mathcal{R}$  (see definition 3.16). The formulas have a level that is not unique (hence the usage of *graphs* instead of *trees*), and is defined in a standard way as being one level greater than the formulas of which they are a direct consequence. In other words, if we use resolution, the level of the resolvent is one level greater than that of its parents (input clauses have level 0).

---

<sup>19</sup> Important works have been carried out on this topic, in particular, by Gödel on second-order logic compared with first-order logic.

The *theorem prover problem* for a triplet:

$$(S_0, \Gamma, F)$$

is defined as the problem of using a strategy  $\Sigma$  to generate a set of formulas  $F$ , with:

$S_0$ : input set;

$\Gamma$ : set of inference rules;

$$\Gamma^*(S_j) = \bigcup_{i \geq j} \Gamma(S_i);$$

$F$ : set of formulas that are subsets of the consequences of  $S_0$  (i.e.  $F \subseteq \Gamma^*(S_0)$ )

and:

$$\Sigma : 2^G \rightarrow 2^G \text{ where } G \text{ is the proof graph.}$$

By unfolding the graph to obtain a tree, we associate to each node a derivation, which permits us to associate a measure of the derivation with strategy  $\Sigma$  to each leaf.

We can slightly modify the definition to also characterize proof assistants, and in particular proof verifiers.

An *abstract proof verifier* is a 5-tuple:

$$(S_0, \Gamma, F, P, \Sigma)$$

where  $P$  is the set of formulas of the alleged proof (if  $P = \emptyset$ , then we have a completely automated theorem prover, if  $P$  contains all the steps of a proof, we have a verifier, if we feed some lemmas to it, we have a proof assistant or an interactive theorem prover). Here, we have included the strategy that is not necessarily uniform: we may think of  $\Sigma$  as a set of strategies. Of course, the theory in which the proof is carried out is contained in  $S_0$ .

DEFINITION 3.19.— A strategy  $st$  for resolution is complete iff for all sets of clauses  $S$ :

$$\text{If } S \vdash_R \square, \text{ then } S \vdash_{R+st} \square.$$

EXAMPLE 3.16.— An example of a strategy for resolution is the *input strategy*: given a set of clauses  $S$ , the resolution rule is always applied with at least one clause in  $S$  (the set of input clauses).  $\square$

EXERCISE 3.39.— a) Give a refutation of the set of clauses  $S$  below:

$$S = \{R, \neg R \vee Q, \neg R \vee S, \neg P \vee \neg Q \vee \neg S, P \vee \neg Q \vee \neg S\}$$

using the input strategy.

b) Is the input strategy complete? Justify.  $\square$

EXERCISE 3.40.– As the goal of the resolution method is to detect an elementary contradiction (i.e. between two unit clauses), an interesting strategy would be to *always* apply the resolution rule with at least one parent clause that is a *unit* clause (if both clauses are unit clauses and the resolution rule can be applied, then we can stop, as we generate  $\square$ ).

This strategy is called the *unit strategy*.

Is this strategy complete? Justify.  $\square$

DEFINITION 3.20.– (complexity of a proof, complexity of a method). *The complexity of a proof (refutation) by resolution of a set of clauses  $S$ , denoted by  $\mathbf{Comp}_{\mathcal{R}}(S)$ , is the number of distinct clauses in the proof (refutation) of  $S$ .*

*The complexity of the resolution method on sets of clauses of cardinality  $n$ , denoted by  $\mathbf{Comp}_{\mathcal{R}}(n)$ , is defined as follows:*

$$\mathbf{Comp}_{\mathcal{R}}(n) = \max_{\text{card}(S)=n} \min_{\mathbf{Comp}_{\mathcal{R}}(S)}$$

The complexity problem of proofs in PL has been studied in detail since the end of the 1960s.

EXERCISE 3.41.– Prove that every set  $S$  containing all  $2^n$  (distinct) clauses of length  $n$  that can be formed using  $n$  propositional symbols is unsatisfiable.  $\square$

EXERCISE 3.42.–

a) Consider a set of  $n$  propositional symbols and let  $p = \lceil \frac{n}{2} \rceil$ , i.e. the smallest integer such that  $p \geq \frac{n}{2}$ .

Prove that the set  $S$  of all positive and negative clauses of length  $p$  that can be formed using  $n$  propositional symbols is unsatisfiable.

b) Does this property still hold if we simply let  $p >_{\min} \frac{n}{2}$  (i.e. the smallest integer that is greater than  $\frac{n}{2}$ )?  $\square$

EXERCISE 3.43.– The *pigeonhole principle* can be stated as follows:

“If we store  $n$  objects ( $n \in \mathbb{N} - \{0, 1\}$ ) in  $n - 1$  boxes, then there is (at least) one box that holds (at least) two objects”, or

“There is no injective application  $\varphi : \{1, 2, \dots, n\} \longrightarrow \{1, 2, \dots, n - 1\}$ ”.

We want to prove (for fixed values of  $n$ ) this principle, using the resolution rule.

Points (a) and (b) below can be swapped.

a) Consider a fixed, arbitrary value of  $n$ , and specify by a *schema*<sup>20</sup> of sets of clauses of PL the set of injective functions from  $\{1, 2, \dots, n\}$  to  $\{1, 2, \dots, n-1\}$ . Let  $S_n$  denote this set of clauses.

b) Fix  $n = 3$  and give  $S_3$ .

c) Can you give a refutation of  $S_3$  in six resolution steps? In ten?  $\square$

EXERCISE 3.44.– Can a set of clauses containing neither any positive clause nor any negative clause be unsatisfiable?

As a corollary to this answer, prove that an unsatisfiable set of clauses contains (at least) *one positive clause and one negative clause*.  $\square$

EXERCISE 3.45.– (three-colorability). Use propositional logic to specify that with three different colours, it is possible to color a map with  $N$  countries, such that each country is colored with a unique color and two neighboring countries are never colored with the same color (there will be maps for which no such coloring is possible). As  $N \in \mathbb{N}$  is not specified, provide a *schema* of the specification.

This is clearly a reduction to the SAT problem. The three-colorability problem is therefore also in NP.

What is the size of this specification? (See also example 9.36.)  $\square$

### 3.9. Horn clauses

These clauses are particularly important in computer science.

DEFINITION 3.21.– (Horn clauses). If  $L, P, L_i$  ( $1 \leq i \leq n$ ) are positive literals, a *Horn clause or conditional formula* is of one of the following forms:

- 1)  $L$
- 2)  $\bigvee_{i=1}^n \neg L_i$
- 3)  $P \vee \bigvee_{i=1}^n \neg L_i$  or  $\bigvee_{i=1}^n (L_i \Rightarrow P)$  or  $(\bigwedge_{i=1}^n L_i) \Rightarrow P$

EXERCISE 3.46.–

a) Prove the following theorem.

If  $H$  is a satisfiable set of Horn clauses then the intersection of any nonempty set of models of  $H$  is also a model of  $H$ .

Horn clauses admit the *model intersection property*.

Note that this theorem also holds for Horn clauses in first-order logic.

b) If we replace “Horn clauses” by “clauses”, does the theorem still hold? Justify.  $\square$

---

<sup>20</sup> Necessarily, as  $n$  is fixed but unknown.

### 3.10. Algebraic point of view of propositional logic

G. Boole introduced the idea that it was possible to treat propositional logic as algebraic expressions in his books *Mathematical Analysis of Logic* and *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. The goals of his research is very well synthesized in the titles (for Boole, *mathematical theories* means “calculus”; in the case of logic “calculus” would now translate into “algebra”).

REMARK 3.41.– In particular, the method proposed by Boole permits us a mathematical treatment of Aristotle’s syllogistic.

It also permits us to compute the probabilities of propositions expressed using non-elementary wffs, and to introduce probabilistic inferences, i.e. inferences that, starting from the probabilities of the premises (events with given probabilities) permit to compute the probability of the conclusions (event whose probability we would like to know). □

REMARK 3.42 (Euler circles, Venn diagrams).– Diagrams are not considered as a part of formal logic, but it has always been acknowledged that they have a great heuristic value and are very useful for informal reasonings.

It suffices to recall how simple it can be to explain the set-theoretic operations of union, intersection, etc., to young children.

Recently, there has been a renewed interest for diagrams in computer science (systems engineering, visual programming, etc.) and in AI (knowledge representation, cognitive and philosophical aspects, etc.).

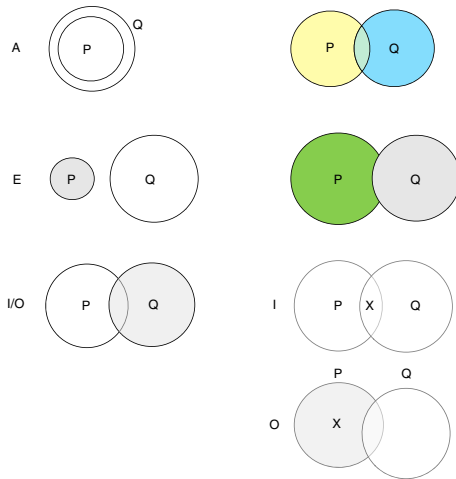
The most famous names associated to different kinds of diagrams are Euler, Venn, L. Carroll, and C.S. Peirce.

John Venn, who admired G. Boole, used diagrams in his book *Symbolic Logic* that have become very popular.

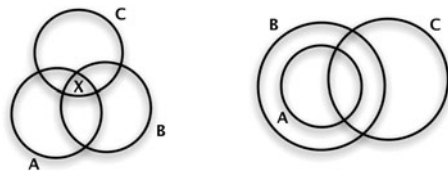
We give examples of Euler circles (left-hand side) and Venn diagrams (right-hand side), together with two syllogisms whose soundness is verified using these diagrams.



Notice that in set theory, contrary to Venn diagrams, the shaded parts represent classes of elements that have a given property.



The last syllogism of example 2.12:



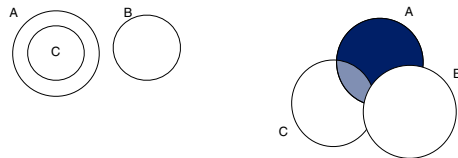
The syllogism:

No A is a B

Every C is an A

---

No C is a B



□

We first recall the definitions that permit us to characterize PL using algebra.

Usually, *lattices* are defined in two separate ways: one as an algebraic structure and the other based on orderings.

**DEFINITION 3.22.**– (Boolean algebra). A *Boolean algebra* is an algebra (see definition 5.4), that is a lattice (T1–T4), which is distributive (T5), bounded (T6), and with complements (T7):

(the operations  $\vee$  and  $\wedge$  are, respectively, called *join* and *meet*.)

$$B = \langle T; \{\vee, \wedge, \bar{\phantom{x}}, 1, 0\} \rangle$$

and for all  $x, y, z$  in  $T$  ( $T$  non-empty, see definition 5.4)

T1)

$$a) x \vee y = y \vee x$$

$$b) x \wedge y = y \wedge x$$

T2)

$$a) x \vee (y \vee z) = (x \vee y) \vee z$$

$$b) x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

T3)

$$a) x \vee x = x$$

$$b) x \wedge x = x$$

T4)

$$a) x \wedge (x \vee y) = x$$

$$b) x \vee (x \wedge y) = x$$

T5)

$$a) x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

$$b) x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$$

T6)

$$a) x \vee 0 = x$$

$$b) x \wedge 1 = x$$

T7)

$$a) x \vee \bar{x} = 1$$

$$b) x \wedge \bar{x} = 0.$$

**EXAMPLE 3.17.**– (set theory, Boolean algebra and PL). Let  $U$  denote a set (universe). The algebra:

$$\langle \mathcal{P}(U); \{\cup, \cap, \bar{\phantom{x}}, U, \emptyset\} \rangle$$

where:

$\mathcal{P}(U)$  : set of subsets of  $U$

is a Boolean algebra.

The well-known relationship between (operators of) set theory and connectives in PL is given in the following table:

$\cup \mapsto \vee$

$\cap \mapsto \wedge$

$\bar{\phantom{x}} \mapsto \neg$

$U \mapsto 1(\mathbf{T})$

$\emptyset \mapsto 0(\mathbf{F})$

More generally, every Boolean algebra is isomorphic to a non-empty family (of subsets of a set) that is closed for the union, intersection, and complement operations<sup>21</sup>.  $\square$

Before giving the definition of a lattice based on the notion of orderings, we recall the definitions of a *partial order*, *sup*, and *inf*.

DEFINITION 3.23.– (order).

– A *partially ordered set* or *poset*  $\langle A, \preceq \rangle$  is a set  $A \neq \emptyset$  and a binary relation  $\preceq$  that satisfies, for all  $x, y, z$  in  $A$ :

- |  |                  |
|--|------------------|
| OP1) $x \preceq x$   | % (reflexivity)  |
| OP2) if $x \preceq y$ and $y \preceq x$ , then $x = y$       | % (antisymmetry) |
| OP3) if $x \preceq y$ and $y \preceq z$ , then $x \preceq z$ | % (transitivity) |

if furthermore:

OP4)  $x \preceq y \text{ or } y \preceq x$

then  $\preceq$  is a total order and  $\langle A, \preceq \rangle$  is a totally ordered set or chain.

(OP1), (OP2), and (OP3) define an order relation and (OP1), (OP2), (OP3), and (OP4) a total order relation.

– Let  $\langle A, \preceq \rangle$  denote a poset and  $H \subseteq A$ ,  $a \in A$  is an upper bound of  $H$  iff for all  $h \in H$   $h \preceq a$ . If for all upper bounds  $b$ ,  $a \preceq b$ , then  $a$  is called the supremum (or least upper bound), denoted by *sup*. The supremum is unique.

---

<sup>21</sup> When  $U$  is finite, this notion corresponds to that of *finite probability spaces*.

Similarly, we define the infimum (greatest lower bound), which is unique and denoted by  $\inf$ .

We define  $x \prec y$  iff  $x \preceq y$  and  $x \neq y$ .

Similarly, we define  $x \preceq y$  iff  $x \prec y$  or  $x = y$ .

An equivalent definition of partially ordered sets follows.

DEFINITION 3.24.– (order-bis).

– A partially ordered set or poset  $\langle A, \prec \rangle$  is a set  $A \neq \emptyset$  together with a binary relation  $\prec$  satisfying, for all  $x, y, z$  in  $A$ :

- |  |                   |
|--|-------------------|
| $OP1'$ ) $x \not\prec x$                                   | % (irreflexivity) |
| $OP2'$ ) if $x \prec y$ then $y \not\prec x$               | % (asymmetry)     |
| $OP3'$ ) if $x \prec y$ and $y \prec z$ , then $x \prec z$ | % (transitivity)  |

Note that  $(OP1')$  and  $(OP3')$  are sufficient to axiomatize an order (see exercise 5.3 d)).

A total order is often defined as follows:

DEFINITION 3.25.– (total order-bis).

– A totally ordered set,  $\langle A, \prec \rangle$  is a set  $A \neq \emptyset$  together with a binary relation  $\prec$  satisfying, for all  $x, y, z$  in  $A$ :

- |   |                   |
|---|-------------------|
| $OT1$ ) $x \not\prec x$                                   | % (irreflexivity) |
| $OT2$ ) if $x \prec y$ and $y \prec z$ , then $x \prec z$ | % (transitivity)  |
| $OT3$ ) $(x \prec y) \vee (x = y) \vee (y \prec x)$       | % (trichotomy)    |

THEOREM 3.6.– A poset  $\langle A, \preceq \rangle$  is a lattice if  $\sup \{a, b\}$  and  $\inf \{a, b\}$  exist for all  $a, b \in A$ .

(We define  $a \vee b := \sup \{a, b\}$ ;  $a \wedge b := \inf \{a, b\}$ )

EXAMPLE 3.18.– The algebra:

$$\langle \{0, 1\}, \{\vee, \wedge, \neg, 1, 0\} \rangle$$

with  $\vee, \wedge, \neg$  defined in definition 3.6 (with  $0: \mathbf{F}$ ,  $1: \mathbf{T}$  and, as in  $\mathbb{N}$ :  $0 \preceq 1$ ) is a Boolean algebra (the simplest one).  $\square$

DEFINITION 3.26.— (congruence, quotient algebra). *An equivalence relation (i.e. a reflexive, symmetric, and transitive relation)  $\sim$  in an algebra  $\langle A; \mathcal{F} \rangle$  is a congruence relation iff for all the operations denoted by the function symbols  $f^{(n)} \in \mathcal{F}$ :*

*if  $a_1 \sim b_1, a_2 \sim b_2, \dots, a_n \sim b_n; a_i, b_i \in A (1 \leq i \leq n)$ , then  $f^{(n)}(a_1, a_2, \dots, a_n) \sim f^{(n)}(b_1, b_2, \dots, b_n)$ .*

*The equivalence classes are denoted by  $| a_i |$ .*

*We can define the operations  $f_{\sim}^{(n)}$  on the partition induced by  $\sim$  (denoted by  $A/\sim$ ):*

$$f_{\sim}^{(n)}(| a_1 |, | a_2 |, \dots, | a_n |) = | f^{(n)}(a_1, a_2, \dots, a_n) |.$$

*The algebra:*

$$\langle A/\sim, \{f_{\sim}^{(n)}\}_{f^{(n)} \in \mathcal{F}} \rangle$$

*is called the quotient algebra.*

If we consider the language  $\mathcal{L}_0$  and we define the binary relation  $\sim$ :

for all  $F, G \in \mathcal{L}_0$   $F \sim G$  iff  $\models F \Leftrightarrow G$

$\sim$  is an equivalence relation.

The equivalence classes thus defined represent the formal definition of a proposition (see also exercise 3.6).

By defining:

$$\begin{aligned} \neg | F | &:^{def} | \neg F | \\ | F | \vee | G | &:^{def} | F \vee G | \\ | F | \wedge | G | &:^{def} | F \wedge G | \\ 0 &:^{def} | F \wedge \neg F | \\ 1 &:^{def} | F \vee \neg F | \end{aligned}$$

the algebra:

$$\mathcal{L}\mathcal{A} = \langle \mathcal{L}_0/\sim, \{\vee, \wedge, \neg, 1, 0\} \rangle$$

is a Boolean algebra, called the Lindenbaum algebra of PL.



## Chapter 4

# First-order Terms

### 4.1. Matching and unification

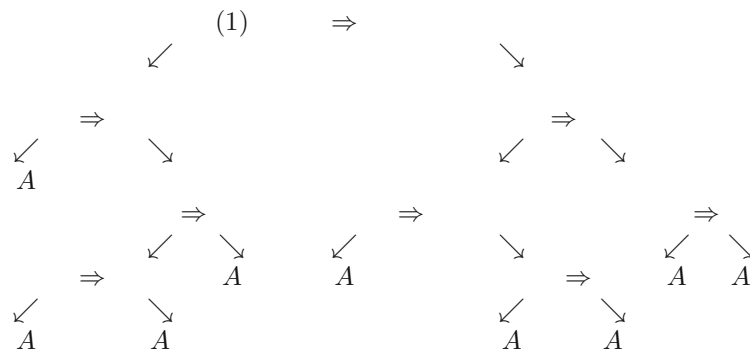
#### 4.1.1. A motivation for searching for a matching algorithm

Imagine that in example 3.9, you are given the first step of an alleged proof:

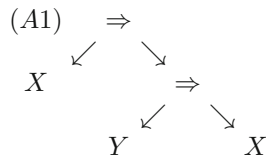
$$1) (A \Rightarrow ((A \Rightarrow A) \Rightarrow A)) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$$

without any justification, and that you (legitimately) wonder: “how can I be sure that this wff is a possible first step of a proof?”.

If formulas (structured strings) are represented as trees, answering this question reduces to finding what replacements should be carried out in the axiom schemas so as to find the desired wff (in a proof, this is the only possibility for the first step). We therefore try the axiom schemas one by one.

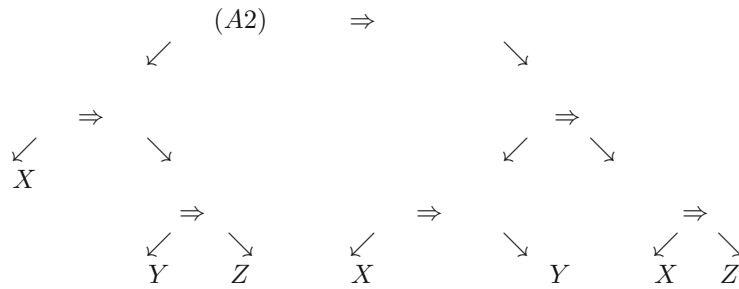


We will use  $X, Y,$  and  $Z$  for the meta-variables that appear in the axiom schemas to better emphasize that they are to be replaced.



It is simple to see that there is no way to replace the variables in (A1) to identify (A1) with (1): we would need to replace  $X$  with  $A$  (rightmost leaf) and  $X$  with  $A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$  (leftmost leaf).

Let us try with (A2):



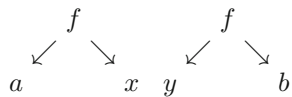
We realize that the trees (1) and (A2) can be identified by replacing:

$$X \leftarrow A; Y \leftarrow A \Rightarrow A; Z \leftarrow A$$

As a conclusion, (1) can indeed be the first step of a proof in  $\mathcal{S}_1$ .

In this example, we assumed that only one of the trees contained variables. Let us see what we would do if both the two terms to be made identical contained variables. In what follows,  $x, y, z, \dots$  denote variables (they are objects that *can be replaced* by other objects) and  $a, b, c, \dots, f, g, \dots$  denote constants (they are objects that *cannot be replaced* by other objects).

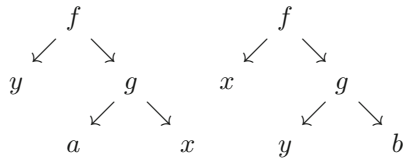
Since we want to design a general algorithm, if  $f, g, \dots$  denote function symbols, then we assume they do not have any particular property (associativity, commutativity, etc.).



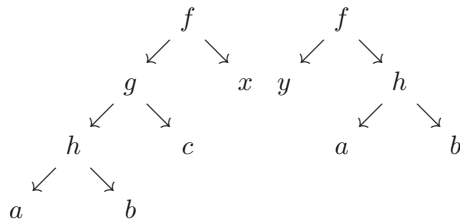
These trees can be made identical as follows:

$$\{x \leftarrow b, y \leftarrow a\}$$





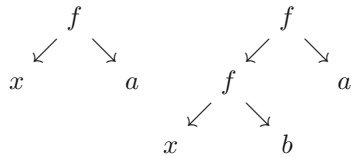
These trees cannot be made identical: we would need to assign  $y \leftarrow a$  and  $y \leftarrow b$ .



These trees can be made identical by assigning:

$$\{y \leftarrow g(h(a, b), c), x \leftarrow h(a, b)\}.$$

Trying to unify the following trees:

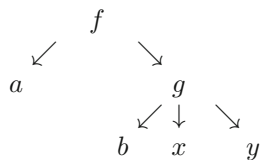


poses a problem. Which one?

#### 4.1.2. A classification of trees

##### Finite trees

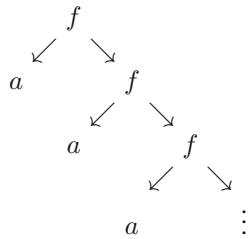
EXAMPLE 4.1.—



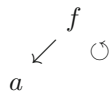
□

Rational infinite trees (i.e. with a finite number of subtrees)

EXAMPLE 4.2.–



i.e.:

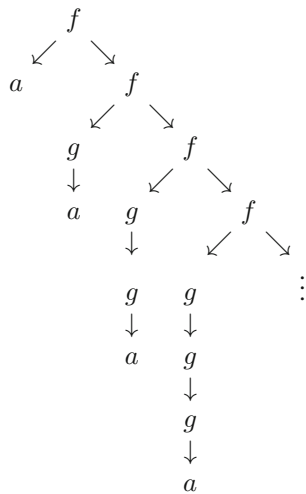


with a linear notation:  $f(a, f(a, f(a, \dots$

□

Non-rational infinite trees

EXAMPLE 4.3.–



with a linear notation:  $f(a, f(g(a), f(g(g(a))), f(\dots$

□

In the next section, we will formalize the concepts that have been introduced in an intuitive way.

## 4.2. First-order terms, substitutions, unification

Let:

$\mathcal{V}$  denote a set of variable symbols;

$$\mathcal{V} = \{x_1, x_2, \dots\};$$

$\mathcal{F}$  denote a set of function symbols, containing *constants* in particular, which are denoted by  $\mathcal{C}$ ,  $\mathcal{C} \subseteq \mathcal{F}$ .

An *arity* ( $n_i \geq 0$ ;  $i \geq 1$ ) is associated with every function symbol (representing the number of its arguments). Constants are of arity 0.

$$\mathcal{F} = \{f_1^{(n_1)}, f_2^{(n_2)}, \dots\}$$

$$(\mathcal{V} \cap \mathcal{F} = \emptyset)$$

**DEFINITION 4.1.**– (terms). *The set of terms constructed on  $\{\mathcal{V}, \mathcal{F}\}$ , denoted by  $\Sigma(\mathcal{V}, \mathcal{F})$ , is the smallest set such that:*

- 1) if  $x \in \mathcal{V}$ , then  $x \in \Sigma(\mathcal{V}, \mathcal{F})$ ;
- 2) if  $a \in \mathcal{C}$ , then  $a \in \Sigma(\mathcal{V}, \mathcal{F})$ ;
- 3) if  $f_k^{(n)} \in \mathcal{F}$  and  $t_1, \dots, t_n \in \Sigma(\mathcal{V}, \mathcal{F})$ , then  $f_k^{(n)}(t_1, \dots, t_n) \in \Sigma(\mathcal{V}, \mathcal{F})$ .

(Note that rule (2) is included in rule (3). It was added here for the sake of clarity).

Terms without any variable (i.e.  $\Sigma(\mathcal{F})$ ) are called *closed terms*.

**REMARK 4.1.**– Infinite trees are not terms.

Unless stated otherwise, we shall note variables  $u, v, x \dots$  and constants  $a, b, c, \dots$   $\square$

**DEFINITION 4.2.**– (variables, constants, depth of a term).

– The set of variables in a term  $t$ , denoted by  $Var(t)$  or  $V(t)$ :

$$Var(t) \text{ (or } V(t)) = \begin{cases} t \in \mathcal{V} & \{t\} \\ t \in \mathcal{C} & \emptyset \\ t = f_k^{(n)}(t_1, \dots, t_n) & \bigcup_{i=1}^n Var(t_i) \end{cases}$$

– The set of constants in a term  $t$ , denoted by  $Const(t)$ :

$$Const(t) = \begin{cases} t \in \mathcal{V} & \emptyset \\ t \in \mathcal{C} & \{t\} \\ t = f_k^{(n)}(t_1, \dots, t_n) & \bigcup_{i=1}^n Const(t_i) \end{cases}$$

– The depth of a term  $t$ , denoted by  $Dpth(t)$ :

$$Dpth(t) = \begin{cases} t \in \mathcal{V} & 1 \\ t \in \mathcal{C} & 1 \\ t = f_k^{(n)}(t_1, \dots, t_n) & 1 + \max\{Dpth(t_1), \dots, Dpth(t_n)\} \end{cases}$$

It was also possible to choose  $Dpth(t) = 0$  in the first two cases.

DEFINITION 4.3.– (substitution).

– A substitution is an application:

$$\sigma : \mathcal{V} \longrightarrow \Sigma(\mathcal{V}, \mathcal{F})$$

which is the identity on all but a finite number of variables.

The domain of a substitution  $\sigma$  is the set:

$$\text{dom}(\sigma) = \{x \mid \sigma(x) \neq x\}$$

and the codomain (or range) of a substitution  $\sigma$  is the set:

$$\text{codom}(\sigma) = \{y \mid \exists x. \sigma(x) = y\}$$

Substitutions are represented by the values of the variables in their domain:

$$\{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \dots, x_n \leftarrow t_n\}$$

or:

$$\{x_1 \mapsto t_1, x_2 \mapsto t_2, \dots, x_n \mapsto t_n\}$$

A substitution

$$\sigma : \mathcal{V} \longrightarrow \Sigma(\mathcal{F})$$

is closed.

Given a substitution

$$\sigma : \mathcal{V} \longrightarrow \Sigma(\mathcal{V}, \mathcal{F})$$

$\sigma$  is extended:

$$\bar{\sigma} : \Sigma(\mathcal{V}, \mathcal{F}) \longrightarrow \Sigma(\mathcal{V}, \mathcal{F})$$

as follows:

$$\bar{\sigma}(t) = \begin{cases} t \in \mathcal{V} & \bar{\sigma}(t) = \sigma(x) \\ t \in \mathcal{C} & \bar{\sigma}(t) = t \\ t = f_k^{(n)}(t_1, \dots, t_n) & \bar{\sigma}(t) = f_k^{(n)}(\bar{\sigma}(t_1), \dots, \bar{\sigma}(t_n)) \end{cases}$$

(It is standard to also denote the extension of  $\sigma$  by  $\bar{\sigma}$ ).

Equality is generally used in the sense of the identity, this is the case, for example, in:

$$(a + b)^2 = a^2 + 2ab + b^2 \text{ (the identity holds for any value of } a \text{ and } b\text{),}$$

or it is used conditionally, this is the case for example in:

$$4 \times x = 16 \times y, \text{ where only some values (possibly none at all) satisfy the equality.}$$

Furthermore, when there are many solutions, some are more interesting than the others. For example, if we are interested in the solutions of the equation above that are positive integers, then  $\{x = 4, y = 1\}$ ,  $\{x = 8, y = 2\}$ ,  $\{x = 12, y = 3\}, \dots$  are solutions.

The most general solution is  $\{x = 4 \times y; y \in \mathbb{N}\}$ .

Notation: to emphasize that this is a conditional equality, we shall write  $4 \times x \doteq 16 \times y$ , and if terms are involved, we shall write  $t_1 \doteq t_2$ .

These comments are a motivation for the following definitions.

**DEFINITION 4.4.**– (ordering on substitutions). *A substitution  $\sigma$  is more general than a substitution  $\gamma$  iff there exists a substitution  $\lambda$  such that  $\gamma = \lambda \circ \sigma$ , where  $\circ$  denotes the composition of substitutions (i.e. of functions).*

**DEFINITION 4.5.**– (unification). *Given the equation  $t_1 \doteq t_2$ , where  $t_1, t_2 \in \Sigma(\mathcal{V}, \mathcal{F})$ , the unification problem consists of finding the most general unifier (mgu) such that:*

$$\sigma(t_1) = \sigma(t_2) \text{ (syntactic identity).}$$

*If only one term contains variables, (say  $t_1$ ), the problem that consists of finding  $\sigma$  such that*

$$\sigma(t_1) = t_2$$

*is the matching problem.*

```

algorithm UNIFICATION
input:  $\mathcal{S} = \{t_1 \doteq s_1, \dots, t_n \doteq s_n\}$ 
output: either a substitution  $\sigma$  (the mgu) solution of  $\mathcal{S}$ 
or  $\perp$  (no solution)
halting test: clash or cycle or no rule applies
% $\Gamma$  denotes a finite set of equations, at the start  $\Gamma = \mathcal{S}$ 
begin
% $V(t)$  and  $V(\Gamma)$  respectively denote the sets of variables in  $t$  and  $\Gamma$ 
% $\Gamma\{x \leftarrow t\}$  means: replace all occurrences of  $x$  in  $\Gamma$  by  $t$ 
% $Expr1 \rightarrow Expr2$  means: replace  $Expr1$  by  $Expr2$ 
For each  $t_i \doteq s_i$  ( $1 \leq i \leq n$ ) in  $\Gamma$  apply the following rules:
1 •  $\{t \doteq t\} \cup \Gamma \rightarrow \Gamma$ 
2 • if  $t$  is not a variable:
 $\{t \doteq x\} \cup \Gamma \rightarrow \{x \doteq t\} \cup \Gamma$ 
3 • if  $x$  is a variable:  $\{x \doteq t\} \cup \{x \doteq s\} \cup \Gamma \rightarrow \{x \doteq t\} \cup \{t \doteq s\} \cup \Gamma$ 
4 •  $\{f(t_1, \dots, t_n) \doteq f(s_1, \dots, s_n)\} \cup \Gamma \rightarrow \{t_1 \doteq s_1\} \cup \dots \cup \{t_n \doteq s_n\} \cup \Gamma$ 
5 • if  $x \notin V(t)$  and  $x \in V(\Gamma)$ :
 $\{x \doteq t\} \cup \Gamma \rightarrow \{x \doteq t\} \cup \Gamma\{x \leftarrow t\}$ 
% for example if  $x = h(y)$ ,  $y = f(u)$  then we must have  $x = h(f(u))$ ,  $y = f(u)$ 
6 •  $\{f(t_1, \dots, t_n) \doteq g(s_1, \dots, s_m)\} \cup \Gamma \rightarrow \perp$  (clash or conflict)
% i.e. function symbols are constants
7 • if  $x \in V(t)$  and  $t$  is not a variable, i.e. if  $t \neq x$ :
% occurs check
 $\{x \doteq t\} \cup \Gamma \rightarrow \perp$  (cycle)
% which means infinite terms are not allowed in  $\sigma$ 
end

```

Figure 4.1. The algorithm UNIFICATION

Notation: we often denote by  $mgu(t_1, t_2)$  the mgu of  $t_1$  and  $t_2$ , and we write  $\sigma t$  instead of  $\sigma(t)$ .

The unification algorithm either constructs the mgu of a set of term equations or detects that there is no solution.

EXERCISE 4.1.–

a) Find the solution (if it exists) of equation:

$$f(x, g(x, y)) \doteq f(g(y, z), g(g(h(u), y), h(u)))$$

b) Find the solution (if it exists) of equation:

$$f(x, f(u, x)) \doteq f(f(y, a), f(z, f(b, z)))$$

c) Given the substitutions

$$\theta = \{x \mapsto a, y \mapsto f(z), z \mapsto x\} \text{ and}$$

$$\sigma = \{x \mapsto b, z \mapsto c, u \mapsto d\}$$

Construct the substitution  $\sigma \circ \theta$ .

d) Consider the equation  $t_1 \doteq t_2$  where  $Var(t_1) \cap Var(t_2) = \emptyset$

Is the **cycle** rule of UNIFICATION still necessary? Could you give a *sufficient* condition guaranteeing the **cycle** rule is not needed?  $\square$

EXERCISE 4.2.– In a calculus called the *equivalential calculus*, the wffs of the language are of the form  $e(X, Y)$ , where  $e$  is a constant symbol (that stands for equivalent), and  $X, Y$  (meta)variables that denote wffs of the language (such as  $A, B, C$  in  $S_1$ ; see section 3.4).

The only inference rule is:

$$CD : \frac{e(A, B) \quad C}{\sigma B} \text{ with } \sigma : mgu(A, C)$$

Question:

Given two wffs

$$e(X, e(X, e(Y, Y))) \text{ and}$$

$$e(Z, Z)$$

1) Can  $CD$  be applied?

2) If so, what are the possibilities?

3) If  $CD$  can be applied, what is(are) the direct consequence(s)?  $\square$

EXERCISE 4.3.– The algorithm UNIFICATION does not assume *any* property on the functions under consideration, and produce a *unique* result (substitution), up to a renaming of the variables.

If we assume that (some of) the binary functions under consideration are commutative, for example:

$$\forall x_1 \forall x_2. f(x_1, x_2) = f(x_2, x_1)$$

Can you modify the algorithm UNIFICATION to take this property into account?

For example, for equation:

$$f(g(a, b), x) \doteq f(h(c), g(y, z))$$

UNIFICATION would return  $\perp$ , but once modified, when  $f$  and  $g$  are commutative, it should produce two solutions:

$$\sigma_1 = \{x \leftarrow h(c), y \leftarrow a, z \leftarrow b\}$$

$$\sigma_2 = \{x \leftarrow h(c), y \leftarrow b, z \leftarrow a\}. \quad \square$$

REMARK 4.2.– Considering function symbols that denote functions with certain properties can turn out to be very complex. For example, if  $f$  denotes an operation (function) that is associative-commutative, i.e.

$$\forall x \forall y. f(x, y) = f(y, x)$$

$$\forall x \forall y \forall z. f(x, f(y, z)) = f(f(x, y), z)$$

then the equation

$$f(a, f(b, z)) \doteq f(x, y)$$

has the following solutions:

$\{x \leftarrow a, y \leftarrow f(b, z)\}, \{x \leftarrow b, y \leftarrow f(a, z)\}, \{x \leftarrow z, y \leftarrow f(a, b)\}, \dots$  and others.

It is simple to verify that we cannot transform one of these unifiers into another by applying a substitution (as previously). We will keep in mind that the solutions are no longer unique (and there can be many of them).  $\square$

The following exercise is an example of how the unification algorithm can be *adapted* to treat data other than terms, by expressing this data as terms and using some *ad hoc* conventions.

EXERCISE 4.4.– Use the algorithm UNIFICATION (that was *modified* in exercise 4.3) to show that, if the unconditional premises (i.e. those that do not contain  $\Rightarrow$ ) are literals, then the rules *MP* (*modus ponens*) and *MT* (*modus tollens*):

$$MP: \frac{A \quad A \Rightarrow B}{B}$$

$$MT: \frac{\neg B \quad A \Rightarrow B}{\neg A}$$

are particular cases of the resolution rule:

$$R: \frac{X \vee \mathcal{X} \quad \neg X \vee \mathcal{Y}}{\mathcal{X} \vee \mathcal{Y}} \quad \square$$



## Chapter 5

# First-Order Logic (FOL) or Predicate Logic (PL1, PC1)

In Aristotle's syllogistic, every statement consists of the attribution of a property to an object, which means that they are expressed using unary predicates and therefore formalized using PL. This is no longer sufficient when we need to consider *relations* (binary, ternary, etc.) between objects. These relations cannot be reduced to unary relations (properties)<sup>1</sup>.

Handling properties and relations is of course essential in programming (logic programming, multi-agent programming, etc.)<sup>2</sup>.

If we believe we can manage using formalizations, such as those after definition 2.8, how can we talk about *objects* that have properties  $P, Q, \dots$  or that are *related* to other objects?

For example, how can we use PL to verify that the following reasoning is correct:

All horses are animals.

Some horses are white.

---

<sup>1</sup> It is interesting to note that in mathematics, a large majority of relations, denoted using predicates, are *binary*.

<sup>2</sup> The *expressive power* of a logic depends on the objects that it permits us to consider (along with the possibility or not of quantifying these objects). The expressive power is obviously related to the *properties of the logic itself* ((un)decidability, etc.). See also section 9.3.

Therefore, some white horses are animals.

Similarly, how can the following reasoning, which should obviously be correct, be translated into PL?

For all  $x$ ,  $xRa$ , therefore there exists an  $x$  such that  $aRx$

If we denote by  $P$ : For all  $x$ ,  $xRa$

and:

$Q$ : there exists  $x$  such that  $aRx$

this reasoning would be classified as *incorrect* ( $P$  can be evaluated to true and  $Q$  to false).

This reasoning can be formalized as follows: (as you already know, see also definition 5.1).

$$\frac{\forall xR(x, a)}{\exists yR(a, y)}$$

and now “we can see” that it is correct.

The soundness of these reasonings is based on the relation between “for all” (or “all”) and “there exists” (or “some”).

Another very familiar example is that of *transitivity*: for any objects named  $x$ ,  $y$ , and  $z$ , if  $x$  is related to  $y$  and  $y$  is related to  $z$  then  $x$  is related to  $z$ . Obviously, these sentences cannot be expressed in PL *in order to be used in a reasoning*. They could be denoted by  $P$  (or  $Q, \dots$ ), but that would not be very useful.

One last example: how can the following sentence be translated in PL?

The object named 0 satisfies property P, and if an object  $x$  satisfies P, then the successor of  $x$  also satisfies P. . .

It is worth mentioning that in *finite* universes, it is possible to stay within PL. For example, the argumentation *if all men are mortal and a given object satisfies the property of being a man, then this object is mortal* (see also example 5.9) could be specified with the following propositional schema in a universe with  $n$  men:

$$\bigwedge_1^n H_i$$

$$\bigwedge_1^n (H_i \Rightarrow M_i)$$

$$H_k \quad \% (1 \leq k \leq n)$$

---


$$M_k \quad \% (1 \leq k \leq n)$$

where the propositional symbols  $H_i$  and  $M_i$ , respectively, denote  $i$  is a man and  $i$  is mortal.

This way of proceeding is not very natural, it is theoretically limited to finite universes and practically limited by the size of these universes; more importantly, it prevents the usage of variables (because all cases have to be enumerated).

We are currently talking about the limits of the expressive power of a language (and how they can be overcome). As can be expected, once more things can be expressed, there is a risk of not being able to answer all questions on wffs of this language with more expressive power.

In computer science, it is well-known that if we use a language that forbids some control structures, then the halting problem becomes decidable for programs written in this language.

From the point of view of decidability: the fragment of FOL called *monadic logic* is decidable (see section 5.8). As soon as predicates with an arity greater than or equal to 2 are allowed, we obtain undecidable fragments.

REMARK 5.1.– The language of FOL was introduced by G. Frege in 1879, but the notations that are currently used are similar to those introduced by Peano in 1889. Formal systems were introduced by Hilbert and Ackermann in 1928 (see also remark 5.21).  $\square$

## 5.1. Syntax

DEFINITION 5.1.– (FOL language). Consider a signature  $\Omega = \{\mathcal{V}, \mathcal{F}, \mathcal{P}\}$ , where:

$\mathcal{V} = \{x_1, x_2, \dots\}$  is a set of variables;

$\mathcal{F} = \{f_1^{n_1}, f_2^{n_2}, \dots\}$  is a set of function symbols,  $n_i$ : arity ( $n_i \geq 0$ ).

Function symbols of arity 0 are constants, and the set of constants is denoted by  $\mathcal{C}$  ( $\mathcal{C} \subseteq \mathcal{F}$ );

$\mathcal{P} = \{=^2, P_1^{k_1}, P_2^{k_2}, \dots\}$  set of predicate symbols,  $k_i$ : arity ( $k_i \geq 1$ );

$\mathcal{V} \cap \mathcal{F} = \mathcal{V} \cap \mathcal{P} = \mathcal{F} \cap \mathcal{P} = \emptyset$ .

The language (i.e. the set of wffs) of FOL, denoted by  $\mathcal{L}_1$ , is the smallest set such that:

- if  $t_1, \dots, t_{k_n} \in \Sigma(\mathcal{V}, \mathcal{F})$  (see definition 4.1), then:  $P_n^{k_n}(t_1, \dots, t_{k_n}) \in \mathcal{L}_1$   
 $P_n^{k_n}(t_1, \dots, t_{k_n})$  is an atomic formula;
- a literal is an atomic formula or its negation;
- if  $\mathcal{P}$  and  $\mathcal{Q} \in \mathcal{L}_1$  and  $x \in \mathcal{V}$ , then:  
 $\neg \mathcal{P} \in \mathcal{L}_1$ ,  $\mathcal{P} \wedge \mathcal{Q} \in \mathcal{L}_1$ ,  $\mathcal{P} \vee \mathcal{Q} \in \mathcal{L}_1$ ,  $\mathcal{P} \Rightarrow \mathcal{Q} \in \mathcal{L}_1$ ,  $\mathcal{P} \Leftrightarrow \mathcal{Q} \in \mathcal{L}_1$ ,  
 $\forall x. \mathcal{P} \in \mathcal{L}_1$ ,  $\exists x. \mathcal{Q} \in \mathcal{L}_1$

(we will also write  $\forall x \mathcal{P}$ ,  $\forall x(\mathcal{P})$ ,  $\exists x \mathcal{Q}$ ,  $\exists x(\mathcal{Q})$ );

–  $\forall, \exists$  are called the universal quantifier and existential quantifier, respectively, and  $x$  is the quantified variable. A variable will not be quantified more than once (that would make no sense);

– in  $\forall x(\mathcal{P})$ ,  $\exists x(\mathcal{Q})$ ,  $\mathcal{P}$  is the scope of the quantifiers.;

– the set  $\{\mathcal{V}, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, =, \forall, \exists\}$  is the set of logical symbols or logical constants;

– the set  $\{\mathcal{F}, \mathcal{C}, \mathcal{P}\}$  is the set of non-logical symbols corresponding to the considered theory;

– if the signature does not contain any functional symbol of arity  $n$  ( $n \geq 0$ ) or the predicate symbol  $=$ , then we have pure FOL;

– a wff can be viewed as a string (or word) on vocabulary  $\Omega$  (i.e. a finite sequence of symbols in  $\Omega$ ). A subformula of a wff  $\mathcal{F}$  is a substring<sup>3</sup> of  $\mathcal{F}$  (considered as a string), which is also a wff;

– a wff is in nnf if it is constructed with literals, the connectives  $\wedge$  and  $\vee$ , and the quantifiers  $\forall$  and  $\exists$ <sup>4</sup>. Every wff can be transformed into another one that is equivalent and in nnf.

REMARK 5.2.– (the set of wffs of FOL is denumerably infinite). As the set of variables is denumerably infinite, the set of logical symbols finite and the set of non-logical symbols denumerably infinite, it is simple to design an algorithm that enumerates all the wffs using the rules of definition 5.1.

Every infinite set of wffs of FOL will therefore be denumerably infinite (as it will be the subset of a denumerably infinite set). □

DIGRESSION 5.1.– In discrete mathematics, researchers use the terminology *Boolean predicates* or *constraints* on Boolean variables.

The arity is the number of variables that occur in the constraints. □

<sup>3</sup>  $w$  is a substring of  $v$  iff there exist strings  $x, y$  (possibly empty) such that  $v = xwy$ .

<sup>4</sup> Thus, if it contains occurrences of  $\neg$ , they are “next” to the atoms.

DIGRESSION 5.2.– (variables 2)<sup>5</sup>. It is interesting to have a historical recollection here on mathematical notations and, in particular, on the notion of a variable.

Letters were used in Antiquity to denote points, lines, etc. in a generic way, meaning that letters were used as *names*.

In arithmetic, symbolic abbreviations are very ancient, they were already used by Egyptians.

Diophantus (circa 3rd Century) introduced a particular symbol for the unknowns in algebra, and this seems to be the very first appearance of what would become a numerical variable in mathematics.

It is acknowledged that the first symbolic algebraic language is due to François Viète (1540–1603). Viète used letters to represent unknowns, powers of the unknowns and undetermined coefficients (*generic names*).

Descartes, Newton, Leibniz, Euler, etc. changed and modified the language introduced by Viète.

In mathematical logic, the concept of a variable was explicitly used by G. Frege. □

DEFINITION 5.2.– (free and bound variables). *The occurrence of a variable  $x$  in a wff of  $\mathcal{L}_1$  is bound iff  $x$  appears immediately after a quantifier, or if  $x$  is in the scope of a quantifier (and has the same name as the quantified variable). Every other occurrence is free.*

*Below,  $\mathcal{P}$  and  $\mathcal{Q}$  denote wffs of  $\mathcal{L}_1$ .*

*Set of free variables in a formula:*

$$Free\_vars(P_n^{k_n}(t_1, \dots, t_{k_n})) = \bigcup_{i=1}^{k_n} Var(t_i) \text{ (see definition 4.2)}$$

$$Free\_vars(\neg \mathcal{P}) = Free\_vars(\mathcal{P})$$

$$Free\_vars(\mathcal{P} \wedge \mathcal{Q}) = Free\_vars(\mathcal{P} \vee \mathcal{Q}) = Free\_vars(\mathcal{P} \Rightarrow \mathcal{Q}) = Free\_vars(\mathcal{P} \Leftrightarrow \mathcal{Q}) = Free\_vars(\mathcal{P}) \cup Free\_vars(\mathcal{Q})$$

$$Free\_vars(\forall x \mathcal{P}) = Free\_vars(\exists x \mathcal{P}) = Free\_vars(\mathcal{P}) \setminus \{x\}$$

---

<sup>5</sup> See also digressions 3.3 and 9.1.

*Set of bound variables in a formula:*

$$\text{Bound\_vars}(P_n^{k_n}(t_1, \dots, t_{k_n})) = \emptyset$$

$$\text{Bound\_vars}(\neg \mathcal{P}) = \text{Bound\_vars}(\mathcal{P})$$

$$\text{Bound\_vars}(\mathcal{P} \wedge \mathcal{Q}) = \text{Bound\_vars}(\mathcal{P} \vee \mathcal{Q}) = \text{Bound\_vars}(\mathcal{P} \Rightarrow \mathcal{Q}) = \text{Bound\_vars}(\mathcal{P} \Leftrightarrow \mathcal{Q}) = \text{Bound\_vars}(\mathcal{P}) \cup \text{Bound\_vars}(\mathcal{Q})$$

$$\text{Bound\_vars}(\forall x \mathcal{P}) = \text{Bound\_vars}(\exists x \mathcal{P}) = \text{Bound\_vars}(\mathcal{P}) \cup \{x\}$$

REMARK 5.3.– The same variable may have free and bound occurrences in a formula, for example, in formula

$$(P(x) \vee \exists y Q(y)) \wedge \forall x (P(x) \vee Q(y))$$

The first occurrence of  $x$  is *free*, the second and third are *bound*.

The first and second occurrences of  $y$  are *bound*, the third is *free*.  $\square$

DEFINITION 5.3.– (closed and open formulas). A wff is *closed*, written *closed wff* (cwff) iff it does not contain any free variable. Otherwise, it is *open* and written *open wff* (owff).

REMARK 5.4.– A bound variable is a variable such that the wff it appears in has a meaning that is *independent* of this variable. For example, in:

$$\int_0^y xy dx$$

$x$  is bound and  $y$  is free.

In programming languages, local variables are bound, global variables are free.

See remark 5.9 on the conceptual difference between a cwff and an owff.  $\square$

EXAMPLE 5.1.– (translation into FOL). Consider the statement:

*Someone who loves all animals loves all men*

that we restate as a reasoning, by formulating all the implicit knowledge. There are two “natural” translations, depending on whether “someone” is translated into:

- i) animal (not necessarily human) or
- ii) human

i.e. respectively:

*Given that every human is also an animal, and that there exists an animal that loves all animals, there exists an animal that loves all men.*

*Given that every human is also an animal and that there exists a human who loves all animals, there exists a human who loves all men.*

We translate both versions by introducing the following predicates:

$A(x)$ :  $x$  is an animal;

$H(x)$ :  $x$  is a human;

$C(x)$ : that must be replaced by  $A(x)$  (case i)), or by  $H(x)$  (case ii)).

The reasoning is represented as follows in FOL:

$$\forall x[H(x) \Rightarrow A(x)]$$

$$\exists y\forall v[C(y) \wedge (A(v) \Rightarrow L(v, y))]$$


---


$$\exists u\forall z[C(u) \wedge (H(z) \Rightarrow L(z, u))]$$

□

## 5.2. Semantics

The concepts of *interpretation*, *model*, and *semantics* that we will define are, similar to the concept of *meaning* in natural language, extremely subtle and deserve some thoughts before providing a formal definition (in FOL).

The notion of a model is essential in logic, computer science, and science generally speaking. It is generally used for many different notions, which is why it is said to be polysemous.

*On the general notion of a model*

Etymologically:

model  $\longrightarrow$  *muid*  $\longrightarrow$  “to prescribe something with authority”.

The word “model” appeared during the 16th Century, meaning measure, musical measure, melody, or way of behaving.

The term is used in different ways in everyday language and in scientific papers. One of the few that can be found in the literature is:

X is a model of Y only if X and Y have the same structure or similar structures.

(This could translate into the existence of a morphism between X and Y)

We can distinguish two meanings that cover many usages:

– representation meant to account for a part of reality that was observed. In general, simplifying hypotheses are allowed, such as idealizations, etc.

A problem that arises involves the limits of the validity of the model, i.e. how can we be sure that all inferences (see Chapter 8) made on the model reflect reality?

A key problem is that of distinguishing between pertinent properties and those that can be discarded. This is of the utmost importance when modeling complex systems, where all the parameters are not well known. Recall the caution of scientists about global warming (models are still not satisfactory).

An example in computer science is that of *models of computation*, such as, computable functions, Turing machines, lambda calculus, Markov algorithms, DNA computing, and quantum computing;

– the one made in mathematical logic. To provide a model of a logical formula means to give an interpretation of the non-logical symbols (i.e. the constant symbols, functions, and predicates) that permits us to make the formula **true**.

Note that this notion of a model can also be applied to the axioms of an empirical theory: we can test “real” configurations of the axiomatization of the theory.

This point of view can be useful (as it is in mathematics) to verify that the axioms of the theory are not contradictory: a model from the logical point of view is a possible realization.

A standard method to better grasp a notion for which there is no formal definition (or a definition that is unanimously accepted) is to enumerate the different ways it is used. This is what we do now.

Some differences in the ways the term is used are clear. There is a clear difference between a scale model or blueprint (bridge, etc.) and a mathematical model of the economy, atom, DNA, and kinetic theory.

Two remarks:

– the design of a model is the basis of any scientific activity in natural science. The goal is to select all (and only) those properties, factors, parameters, etc., that are supposed to be pertinent for the studied phenomena;

– the distinction between a *theory* and a *model* is not always clear. In general, a model is considered as a step toward a theory.

The notion of a morphism that we mentioned above can be better understood by considering phenomena that are (essentially) described by the same formulas: oscillation of a circuit or a pendulum, flow of a liquid or flow of an electric charge, etc.



Models can have different *functions*.

For example, in biology, models can have a *normative* function (see the etymology of the word model at the beginning of this section) to organize data. The capital role here is to classify valid inferences (meaning putting information in a usable form (see Chapter 8 for the definition of this term).

It can also be *explanatory* (in physics, medicine, etc.) or *educational* (such as in science popularization TV shows), *prospective* (social sciences, ecology, etc.), *heuristic* (computer science, biology, etc.), *descriptive* (simulation), etc.

On the importance of the notion of a model in science, we quote an opinion of one of the greatest mathematicians of the 20th Century (S. Mac Lane):

... the sciences do not try to explain, they hardly try to interpret, they mainly make models... The justification (of a model) is solely and precisely that it is expected to work ... Furthermore, it must satisfy certain aesthetic criteria – that is, in relation to how much it describes, it must be simple.

### 5.2.1. *The notions of truth and satisfaction*

What is truth?

Pilate (Gospel according to John, 18:38)

What I say three times is true

L. Carrol (The Hunting of the Snark)

The notion of semantics (with the meaning of study of a language, i.e. its words and statements from the point of view of their *meaning*) is very difficult to specify, and our intuition seems to associate it with the notion of *translation*.

The notions of **true** and **false** are closely related to that of meaning.

To give a characterization of the notion of truth is an old problem of philosophy and logic<sup>6</sup>.

As Tarski noticed, the word truth and those derived from it are used in everyday language in different domains: psychology (for example, “Does Ann really love Bernard?”), aestheticism (for example, “Is this book really a masterpiece?”), moral (for

---

<sup>6</sup> In the Western world, since Parmenides and Aristotle.

example, “Why isn’t this politician telling the truth?”), religion (for example, “Did this miracle really happen?”), etc.

For some, statements are the vehicles of veracity or falsity, for others, meanings are.

Once we agree on this, there remains to decide how to assign them with a truth value (or what their truth value is).

There is, however, a large consensus on the following: *what is true or false<sup>7</sup> are the propositions* (see definition 3.1).

A great philosopher (and logician) said it this way: “The eternal statements (i.e. independent from time<sup>8</sup>) are what I consider essentially as vehicles of truth”.

But what does the truth of these statements consist in? They qualify as true (according to Tarski) depending on reality.

“Snow is white” iff snow is white (in other words, truth is *disquotation*).

The truth predicate is an in-between, between words and statements and the real world. What is true is the statement, but its truth holds because the world is as it states.

By analyzing paradoxes, we can conclude that the truth predicate is incoherent, unless it is somehow restricted (there exists a theorem by Tarski on the undefinability of truth).

Basically, the problem is solved by putting a hierarchy in place, (similar to what was done in set theory):  $\text{true}_0$  disquotes all statements without any truth predicate;  $\text{true}_1$  disquotes all statements without any truth predicate above  $\text{true}_0$ ... and so on.

Here, we restrict ourselves to the notion of truth in axiomatic systems (mathematical logic) about (well-formed) formulas of a formal language.

Before proceeding with the formalization, a question arises naturally: “What criteria should be satisfied by a suitable definition of truth?”

A. Tarski gave the following three criteria:

---

<sup>7</sup> The notion of truth is closely related to that of negation by the equation **false = not true**.

<sup>8</sup> This is not the case, for example, with the statement “my head hurts”.

1) Meta-language: if  $L$  is the object language (working language), then the definition of truth must be given in a meta-language  $M$  with  $L \subset M$ ,  $M$  must globally be capable of expressing “more things” than  $L$ . If  $L$  is capable of expressing facts about its own semantics, then we easily obtain paradoxes such as the liar’s paradox.

$M$  contains a unary predicate **True** whose intended meaning is: **True(prop)**: proposition **prop** is true;

2) Formal correction: predicate **True** must be of the form (or in a form provably equivalent to):

$$(\star) \text{True}(x) \Leftrightarrow \Psi[x] \text{ ' } x \notin \text{Var}(\Psi[x])$$

If an equivalent form is used, the equivalence must be proved using the axioms of  $M$ , but it must not use the predicate **True**;

3) Material adequation: the objects that satisfy definition  $(\star)$  in (2) must be exactly those that are intuitively true propositions in  $L$ , and this must be proved using the axioms of  $M$ .

To avoid problems related to the definition of truth, Tarski introduced the *satisfaction* relation (see definition 5.6). The key to avoiding problems is that the definition of satisfaction is *inductive* (it is said to be *compositional*): we begin with the satisfaction of atomic statements, then the satisfaction of statements of a higher complexity is defined in terms of that of their components (the idea of a hierarchy can be found here too). Truth only deals with closed statements (which do not contain free variables). The analog to *truth* for open statements is *satisfaction*. An assignment of objects satisfies a statement if the latter is true for the values given to the free variables<sup>9</sup>. The notion of satisfaction *does not permit* us to translate, for example, “not ( $x$  satisfies  $x$ )” (“ $x$ ” denotes a variable).

To get a better grasp of the idea behind the formal definition, we first give a few informal definitions.

EXAMPLE 5.2.— We consider a “formal system” (see definition 3.9)<sup>10</sup>  $\mathcal{S} = \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$

where:

$\mathcal{L}$ : English language;

$\mathcal{R}$ : the “usual” rules of mathematics;

<sup>9</sup> A closed statement is satisfied either by all assignments or by none, depending on whether it is true or false.

<sup>10</sup> According to our definition, this is actually not a formal system: a *formal* language and *formal* inference rules are missing, but the context is clear enough and there is no ambiguity.

$\mathcal{A}$ :

let  $K$  and  $L$  be two sets

A1): every element of  $L$  contains exactly two elements of  $K$ .

A2): no element of  $K$  is contained in more than two elements of  $L$ .

A3): no element of  $L$  contains all the elements of  $K$ .

A4): the intersection of any two (distinct) elements of  $L$  contains exactly one element of  $K$ .

A possible *model, interpretation, and meaning* of these axioms (that also shows that these axioms are not contradictory) is:

$$K = \{A, B, C\}$$

$$L = \{\{A, B\}, \{B, C\}, \{C, A\}\}$$

(this model can be viewed as a triangle with the (non-collinear) vertices  $A, B, C$ ).  $\square$

EXAMPLE 5.3.– (Z. Manna and R. Waldinger). When learning algorithm, it turns out that program *schemas* frequently occur in different problems.

Consider the following schema, for which we will produce different interpretations.

```

program X;
begin
  read ( $x$ ) ; %  $x$  is a variable
   $y_1 \leftarrow x$  ;
   $y_2 \leftarrow a$  ; %  $a$  is a constant
  while  $\neg P(y_1)$ 
  do
     $y_2 \leftarrow g(y_1, y_2)$  ;
     $y_1 \leftarrow f(y_1)$  ;
  enddo
   $z \leftarrow y_2$  %  $z$  contains the result
end

```

– **Interpretation 1**

$D$  (considered universe):  $\mathbb{N}$

$a$ : 1

$f(y_1)$ :  $y_1 - 1$

$$g(y_1, y_2): y_1 \times y_2$$

$$P(y_1): y_1 = 0$$

The program represents factorial (x)

– **Interpretation 2**

$D$  (considered universe): lists

$a$ : nil

$f(y_1)$ : *cdr*( $y_1$ ) %, i.e. list  $y_1$  without its first element

$g(y_1, y_2)$ : *cons*(*car*( $y_1$ ),  $y_2$ ) %, i.e. the list obtained by putting the first element of  $y_1$  instead of that of  $y_2$

$P(y_1)$ : *null*( $y_1$ ) % *null*( $y$ ):  $y$  is an empty list

The program represents reverse (x)

– **Interpretation 3**

$D$  (considered universe):  $\mathbb{N}$

$a$ : 0

$f(y_1)$ :  $y_1 - 1$

$g(y_1, y_2)$ :  $y_1 + y_2$

$P(y_1)$ :  $y_1 = 0$

The program represents the sum of the first x natural numbers. □

EXAMPLE 5.4.– Consider the following wffs:

a)  $\forall x (\neg P(x, x) \wedge (\forall y \forall z ((P(x, y) \wedge P(y, z)) \Rightarrow P(x, z))) \wedge \exists w P(x, w))$

This could express (for example): no natural number is less than itself, the relation “less than” is transitive and there always exists a natural number that is greater than a given natural number.

b)  $\forall y \exists x P(x, y)$

This could express (for example): every integer is greater than some other integer.

c)  $\forall y \exists x P(x, y)$

We could say this is false if we consider natural numbers and  $P$  represents the relation “less than”.

d)  $\forall x \neg D(a, x)$

This could express (for example): 0 does not divide any integer.

e)  $\forall x P(f(x))$

This could express (for example): the square of any integer is positive.

$$\text{f) } \forall x \forall y (P(x, y) \Rightarrow P(x, y))$$

This could express (for example): for any relation we could imagine that is represented by  $P$ , this wff is true.

$$\text{g) } \exists x \exists y \neg (P(x, y) \Rightarrow P(x, y))$$

This could express: for any relation we could imagine that is represented by  $P$ , this wff is false.

Consider the following wff:

$$\text{h) } \forall x \exists y P(x, y)$$

is it true or false? The (correct) answer that seems the most natural is, it depends. Indeed, if the formula is interpreted on  $\mathbb{N}$  and  $P(x, y)$  represents the relation  $x \leq y$ , then it is true. If  $P(x, y)$  represents  $x > y$ , then it is false.

$$\text{i) } \forall x \exists y (P(x, y) \wedge \forall w (P(y, w) \Rightarrow y = w))$$

This could represent: if the values of variables  $x, y$  and  $w$  are instants and  $P(x, y)$  denotes the relation  $x \leq y$  between these instants, the formula expresses that there will be an end of time.  $\square$

**DEFINITION 5.4.**— Given a first-order language  $\mathcal{L}_1$  (see definition 5.1), a first-order structure or structure  $\mathcal{M}$  is a triplet:

$$\mathcal{M} = \langle D; \mathcal{F}, \mathcal{R} \rangle$$

where:

$D$  is a non-empty set called the domain or universe of discourse

$$\mathcal{F} = \{f_1^{(i_1)}, \dots, f_n^{(i_n)}, \dots\}$$

set of functions:

$$f_j^{(i_j)}: D^{i_j} \longrightarrow D$$

$$\mathcal{R} = \{r_1^{(k_1)}, \dots, r_n^{(k_n)}, \dots\}$$

set of relations:

$$r_j^{(k_j)} \subseteq D^{k_j}$$

where the  $f_j^{(i_j)}$ 's (in particular, the constants) and the  $r_j^{(k_j)}$ 's, respectively, correspond to the functional symbols and the predicates in  $\mathcal{L}_1$ .

Two particular cases:

$\mathcal{M} = \langle D; \mathcal{F} \rangle$ : (abstract) algebra

$\mathcal{M} = \langle D; \mathcal{R} \rangle$ : relational system

EXAMPLE 5.5.–

$\mathcal{M}_1 = \langle \mathbf{Z}; \{+\}, \{\leq\} \rangle$ : structure

$\mathcal{M}_2 = \langle \mathbf{Z}; \{\leq\} \rangle$ : relational system

$\mathcal{M}_3 = \langle \mathbf{R}; \{+, -, \times\} \rangle$ : algebra □

DIGRESSION 5.3.– When searching for interpretations of wffs or sets of wffs, it is often the case that different interpretations are “globally” the same.

More precisely, to each element of one domain corresponds an element of the other domain with the same properties and conversely.

This feature is formalized in the following definition. □

DEFINITION 5.5.– (structure isomorphism). *Given two structures  $\mathcal{M}_1$  and  $\mathcal{M}_2$ :*

$$\mathcal{M}_1 = \langle D_1; \mathcal{F}_1, \mathcal{R}_1 \rangle$$

$$\mathcal{M}_2 = \langle D_2; \mathcal{F}_2, \mathcal{R}_2 \rangle$$

*a bijection  $\mathcal{I}: D_1 \rightarrow D_2$*

*is a structure isomorphism iff*

(the exponents  $\mathcal{M}_1$  and  $\mathcal{M}_2$  identify the structure the objects belong to)

– for every constant  $c$  in the signature of  $\mathcal{L}_1$ :

$$\mathcal{I}(c^{\mathcal{M}_1}) = c^{\mathcal{M}_2}$$

– for every  $n$ -tuple  $(d_1, d_2, \dots, d_n) \in D_1^n$  and every  $n$ -ary functional symbol  $f^{(n)}$  in the signature of  $\mathcal{L}_1$ :

$$\mathcal{I}(f^{(n)\mathcal{M}_1}(d_1, d_2, \dots, d_n)) = f^{(n)\mathcal{M}_2}(\mathcal{I}(d_1), \mathcal{I}(d_2), \dots, \mathcal{I}(d_n))$$

– for every  $n$ -tuple  $(d_1, d_2, \dots, d_n) \in D_1^n$  and every  $n$ -ary predicate symbol  $P^{(n)}$  in the signature of  $\mathcal{L}_1$ :

$$P^{(n)\mathcal{M}_1}(d_1, d_2, \dots, d_n) = P^{(n)\mathcal{M}_2}(\mathcal{I}(d_1), \mathcal{I}(d_2), \dots, \mathcal{I}(d_n))$$

$\mathcal{M}_1$  and  $\mathcal{M}_2$  are said to be isomorphic.

REMARK 5.5.– The idea behind the following definition is the same as the one that we used informally and intuitively in the examples, i.e. given a first-order language (or wff), we fix a *universe*  $D$ , to each constant in the wff we associate an element

of  $D$ , to each functional symbol of arity  $n$ , a *total* function (meaning that it is defined everywhere on  $D$ ) of arity  $n$  on  $D$ , and to each predicate symbol of arity  $k$ , a relation of arity  $k$  on  $D$ . Variables “go through”  $D$ , and  $\forall$  and  $\exists$  are interpreted as usual, i.e. “for all” and “there exists (at least one),” respectively. The semantics (meaning) of the wff is given by that of the functions and relations on the chosen universe, which are assumed to be known.  $\square$

A cwff is **T** or **F** in an interpretation. The evaluation of the cwff is carried out in the structure and the variables get their values in the universe of discourse. It is, therefore, not necessary to add to the language expressions of the form “for every  $x$  in  $D$ ”. The language is independent of the formalization of set theory.

**DEFINITION 5.6.**– (interpretation, satisfaction). *Let  $\mathcal{L}_1$  denote a first-order language and  $\mathcal{M} = \langle D; \mathcal{F}, \mathcal{R} \rangle$  a structure.*

*An fp-assignment ( $f$  stands for “function” and  $p$  stands for “predicate”) is a function  $a_{fp}$  satisfying the following:*

*i) for every predicate symbol  $P_k^{(k_n)}$ ,  $a_{fp}(P_k^{(k_n)})$  is a relation  $P_k^{\mathcal{M}} \subseteq D^{k_n}$  of  $\mathcal{R}$ ;*

*ii) for every function symbol  $f_j^{(j_n)}$ :*

*$a_{fp}(f_j^{(j_n)})$  is a (total) function  $f_j^{\mathcal{M}} : D^{j_n} \rightarrow D$  of  $\mathcal{F}$*

*for constants :*

*$a_{fp}(a) = a$  with  $a \in D$*

*a v-assignment is a function:*

*$a_v : \mathcal{V} \rightarrow D$*

*where  $\mathcal{V}$  denotes the set of variables of  $\mathcal{L}_1$ .*

*For terms (see definition 4.1) we define the t-assignment  $a_t$  as follows:*

*iii) if  $t \in \mathcal{V}$  then  $a_t(t) = a_v(t)$ ;*

*iv) if  $t \in \mathcal{C}$  then  $a_t(t) = a_{fp}(t)$ ;*

*v) otherwise,  $a_t(f_j^{(j_n)}(t_1, \dots, t_{j_n})) = f_j^{\mathcal{M}}(a_t(t_1), \dots, a_t(t_{j_n}))$ .*

*– An interpretation of the language  $\mathcal{L}_1$  (or of a wff of  $\mathcal{L}_1$ ) is an fp, v, and t-assignment.*

*– A satisfaction relation:*

*$\mathcal{M} \models_{\mathcal{I}} \varphi$  (which reads as “Formula  $\varphi$  is satisfied in structure  $\mathcal{M}$  with interpretation  $\mathcal{I}$ ”)*



is defined as follows:

- 1)  $\mathcal{M} \models_{\mathcal{I}} P_k^{(k_n)}(t_1, \dots, t_{k_n})$  iff  $(t_1, \dots, t_{k_n}) \in P_k^{\mathcal{M}}$
- 2)  $\mathcal{M} \models_{\mathcal{I}} \neg\varphi$  iff not  $\mathcal{M} \models_{\mathcal{I}} \varphi$  %, i.e. logic with two truth values
- 3)  $\mathcal{M} \models_{\mathcal{I}} \varphi \wedge \psi$  iff  $\mathcal{M} \models_{\mathcal{I}} \varphi$  and  $\mathcal{M} \models_{\mathcal{I}} \psi$
- 4)  $\mathcal{M} \models_{\mathcal{I}} \varphi \vee \psi$  iff  $\mathcal{M} \models_{\mathcal{I}} \varphi$  or  $\mathcal{M} \models_{\mathcal{I}} \psi$
- 5)  $\mathcal{M} \models_{\mathcal{I}} \varphi \Rightarrow \psi$  iff (not  $\mathcal{M} \models_{\mathcal{I}} \varphi$ ) or  $\mathcal{M} \models_{\mathcal{I}} \psi$
- 6)  $\mathcal{M} \models_{\mathcal{I}} \varphi \Leftrightarrow \psi$  iff  $\mathcal{M} \models_{\mathcal{I}} \varphi \Rightarrow \psi$  and  $\mathcal{M} \models_{\mathcal{I}} \psi \Rightarrow \varphi$
- 7)  $\mathcal{M} \models_{\mathcal{I}} t_1 = t_2$  iff  $t_1^{\mathcal{M}} = t_2^{\mathcal{M}}$  % see section 9.1
- 8)  $\mathcal{M} \models_{\mathcal{I}} \exists x\varphi$  iff there exists  $a \in D$  such that  $\mathcal{M} \models_{\mathcal{I}[x|a]} \varphi$
- 9)  $\mathcal{M} \models_{\mathcal{I}} \forall x\varphi$  iff for all  $a \in D$   $\mathcal{M} \models_{\mathcal{I}[x|a]} \varphi$

$\mathcal{I}[x | a]$  coincides with  $\mathcal{I}$  except for  $a_v(x)$

REMARK 5.6.– Warning: not  $(\phi \models \varphi)$  is not equivalent to  $\phi \models \neg\varphi$ .

See section 3.1.1 and remark 3.5. The negation of “for all” is not “for all not”.  $\square$

REMARK 5.7.– (on the domain of discourse). In the definition of the semantics for FOL, the only constraint on the domain of discourse is that it is non-empty. We can therefore choose the domain of discourse to be a set of closed terms. This remark will be useful in the proof of theorem 5.4.  $\square$

EXAMPLE 5.6.– The wff

$$\forall x \exists y \exists z (x + z = y \wedge \exists w (w + w = y))$$

denotes in what is called Presburger arithmetic, i.e. the theory of the structure:

$$\mathbb{N}_A = \langle \mathbb{N}; 0, succ, +, =, < \rangle$$

“There are infinitely many even numbers”.

This theory is decidable, but the decision procedure has a superexponential complexity ( $2^{2^n}$ ).  $\square$

DEFINITION 5.7.– A wff  $\varphi$  is valid iff it is satisfied in every interpretation on every structure. This is denoted by  $\models \varphi$ .

A structure  $\mathcal{M}$  is a model of a set of wffs  $S$  iff there exists  $\mathcal{I}$  such that  $\mathcal{M} \models_{\mathcal{I}} \varphi$  for all  $\varphi \in S$ .

REMARK 5.8.– (models: another theory). A model of an axiomatic theory is a set of objects chosen from another theory: the one in which the objects assigned to those of the former are supposed to have a meaning “by themselves” (for example, sets, functions, and relations) and satisfying the axioms.  $\square$

REMARK 5.9.– (open and closed formulas).

– A cwff denotes a *truth value*. For example,  $\forall x \exists y P(x, y)$  denotes **true** in  $\mathbb{N}$ , if  $P$  represents  $<$ .

– An owff denotes a *set*, i.e. the set of values that make it true. For example,  $Prime(x)$  denotes all prime numbers.

– Owffs are used to define classes (sets) (see digression 2.2), for example, the set  $\{x \mid Prime(x)\}$ .

– Free variables, and therefore owffs, occur very frequently in informal mathematics, in expressions such as:

i) *Let  $x$  denote a natural number*

or :

ii) *Let  $x$  denote a natural number such that ...*

In (i) they are given a universal interpretation (i.e. *for all  $x$* ).

In (ii) they are given a conditional interpretation (i.e. *for some particular  $x$ 's*).

When these expressions (or rather their formalizations) are involved in reasonings, they will be treated, as is customary, as *universally quantified* variables. Indeed, when no particular conditions are imposed on  $x$ , this is equivalent to saying *for any  $x$* , or *for all  $x$* .

EXAMPLE 5.7.– (set of prime numbers). The set of prime numbers in  $\mathbb{N}$  can be defined as follows:

$$Prime(x) = \{x \neq 1 \wedge \forall y \forall z ((x = y \times z) \Rightarrow (y = 1 \vee z = 1))\}. \quad \square$$

When a condition is imposed on  $x$ , this is equivalent to saying *for all  $x$  satisfying this condition* (condition that will be translated in the formula)<sup>11</sup>.

– For the owffs that occur in reasonings, we shall use their closure, which is defined by:

$$cwf(F) \stackrel{def}{=} \text{if } F : \text{cwff then } F \text{ else } \forall x_1 \forall x_2 \dots \forall x_n F$$

(where  $Free\_vars(F) = \{x_1, x_2, \dots, x_n\}$ ). □

EXERCISE 5.1.– (A model is *finite* (respectively, *infinite*) if the cardinality of its universe of discourse is finite (respectively, infinite).

---

<sup>11</sup> For example, the expression *Let  $x$  denote a prime number* translates into  $\forall x(x \in \mathbb{N} \wedge Prime(x))$ .

a) Give a model of:

$$\forall x P(g(x, f(x)), a)$$

b) Give a model and a counter model of:

$$\forall x \forall y (P(f(x, y), a) \Rightarrow P(x, y))$$

c) Give a model of:

$$\forall x \forall y \forall z (P(x, y) \Rightarrow P(f(x, z), f(y, z)))$$

d) Give a model of:

$$\forall x \exists y P(x, f(f(x, y), y))$$

e) Give a model of:

$$\forall x \forall y (P(f(x, a), y) \Rightarrow P(f(y, a), x))$$

f) Give a counter model of:

$$\forall x \exists y P(x, y) \Rightarrow \exists y \forall x P(x, y)$$

g) Give a model and a counter model of:

$$\forall x \forall y [(P(x, y) \wedge \neg Q(x, y)) \Rightarrow (\exists z (P(x, z) \wedge \neg Q(z, x) \wedge P(z, y) \wedge \neg Q(z, y)))]$$

h) Is the wff:

$$\forall x (P(x) \vee Q(x)) \Rightarrow \forall x P(x) \vee \forall x Q(x)$$

valid?

i) Give a model of the following formula that is *independent* of the selected domain of discourse.

In other words, give an interpretation of the predicate  $P$  such that for any domain, it corresponds to a model of the given formula:

$$\forall x \forall y \forall z. P(x, y) \wedge P(x, z) \Leftrightarrow P(x, y) \wedge P(y, z)$$

j) Given the wff:

$$\forall x \forall y \forall z (P(x, y) \vee P(y, z) \vee P(x, z))$$

$j_1$ ) give a model of this formula with domain  $D = \mathbb{R}$  (i.e. an *infinite* model);

$j_2$ ) give a model of this formula with domain  $D = \{1, 2\}$  (i.e. a *finite* domain);

$j_3$ ) give a counter model of this formula with domain  $D = \mathbb{N}$ .

k) Can you construct:

$k_1$ ) a finite model for the following formula?

$$\forall x \neg P(x, x) \wedge \forall x \exists y P(x, y) \wedge \forall x \forall y \forall z (P(x, y) \wedge P(y, z) \Rightarrow P(x, z))$$

$k_2$ ) an uncountably infinite model?

$k_3$ ) a denumerably infinite model?

l) Give a model and a counter model of:

$$\forall x \exists y \exists z \forall u [\neg E(y, z) \wedge A(x, y) \wedge A(x, z) \wedge A(x, u)] \Rightarrow (E(u, y) \vee E(u, z))$$

m) Can you construct:

$m_1$ ) an infinite model for the formula:

$$\forall x \exists y \exists z (f(y) = x \wedge f(z) = x \wedge y \neq z) ?$$

$m_2$ ) a finite model?

n) Can you construct a finite model for the formula:

$$\forall x \forall y ((f(x) = f(y)) \Rightarrow (x = y)) \wedge \exists x \forall y f(y) \neq x$$

i.e. a function  $f : D \rightarrow D$  that is one-to-one but not onto.

o) Given the formulas (1), (2), and (3) below, show that (3) is not a logical consequence of (1) and (2) (in other words, the commutativity of  $+$  cannot be deduced from (1) and (2))

$$1) \forall x. 0 + x = x$$

$$2) \forall x \forall y. s(x) + y = s(x + y)$$

$$3) \forall x. x + 0 = x$$

0 denotes a constant,  $x$  and  $y$  variables and  $s$  a functional symbol.

p) Can you construct a model for the set of formulas below?

$$\forall x. f(a, x) = x$$

$$\forall x. f(s(x), y) = s(f(x, y))$$

$$\forall x. s(p(x)) = x$$

$$\forall x. p(s(x)) = x$$

$$\forall x. f(p(x), y) = p(f(x, y))$$

$a$  denotes a constant,  $x$  and  $y$  variables, and  $f$ ,  $s$ , and  $p$  functional symbols. □

### 5.2.2. A variant: multi-sorted structures

In section 5.2.1, it was mentioned that it was not necessary to specify what domain a variable can get its values from in a formula.

However, it is common in practice to want to identify these domains (sets that are named *sorts*), in particular, when different variables can get their values from domains of different types (for example, scalars and vectors in vector spaces).

#### 5.2.2.1. Expressive power, sort reduction

We may wonder whether using sorts increases the expressive power of FOL, i.e. if more things can be expressed with sorts than without or if this is simply syntactic sugar. The answer is that the expressive power of FOL with sorts is the same as that of FOL, and this result is proved using the so-called sort reduction technique.

$$\mathcal{M} = \langle D, E ; \mathcal{F}, \mathcal{R} \rangle$$

– The signature is extended with predicates  $P_D$  and  $P_E$ .

– We consider the structure

$$\mathcal{M}' = \langle D \cup E; \mathcal{F}, \mathcal{R} \cup \{R_D, R_E\} \rangle$$

where  $R_D = D$ ,  $R_E = E$  ( $R_D$  and  $R_E$ , are unary relations, i.e. subsets of  $D$  and  $E$ , respectively) and:

$$a_{fp}(P_D) = R_D \text{ and } a_{fp}(P_E) = R_E \text{ (see definition 5.6).}$$

– Hence, if sorts were used to write formulas, it suffices to replace (as usual,  $F[x]$  means that variable  $x$  occurs in wff  $F$ ):

$$\forall x \in D.F[x] \text{ by } \forall x.P_D(x) \Rightarrow F[x]$$

$$\forall x \in E.F[x] \text{ by } \forall x.P_E(x) \Rightarrow F[x]$$

$$\exists x \in D.F[x] \text{ by } \exists x.P_D(x) \wedge F[x]$$

$$\exists x \in E.F[x] \text{ by } \exists x.P_E(x) \wedge F[x]$$

REMARK 5.10.– (theory, closed theory, and complete theory). In definition 3.9, we defined the notion of a *formal theory*. Here, we introduce the notion of a *theory*, along with essential definitions and properties of theories, that deal with semantics.

– Once the notion of *consequence* has been introduced (see definitions 2.4 and 2.6, it can be syntactic or semantic), a theory is closed iff it is closed by the consequence relation (according to Gödel's completeness theorem, see remark 5.21, we can use  $\vdash$  or  $\models$ ).

– A theory  $T$  is complete (in  $\mathcal{L}_1$ ) iff the set of its consequences is maximal consistent (i.e. if it is consistent and none of its supersets is consistent).

– A set of axioms of a theory  $T$  is a set of cwffs with the same set of consequences as  $T$ .  $T$  is a set of axioms of  $T$  and  $\emptyset$  is a set of axioms of  $T$  iff  $T$  is the set of valid cwffs of  $\mathcal{L}_1$  (from a semantical point of view).  $T$  is finitely axiomatizable iff it is axiomatizable by a finite set of axioms.

– (See remark 3.15) The standard way of specifying a theory is to provide a set of axioms that define it.

Here is another way of proceeding: given a structure  $\mathcal{M}$  and an interpretation  $\mathcal{I}$  of  $\mathcal{L}_1$ , the theory  $\mathcal{M}$  is the set of all cwffs  $F$  of  $\mathcal{L}_1$  such that  $\mathcal{M} \models_{\mathcal{I}} F$ . Theory  $\mathcal{M}$  is complete.

– A theory  $T$  is complete iff for all cwffs  $F$ , either  $T \models F$  or  $T \models \neg F$ . % This is to be compared with the notion of completeness for a *formal system* (definition 3.12).

Given a theory  $T$ , statements 1 to 4 below are equivalent:

- 1) the set of consequences of  $T$  is maximal consistent;
- 2)  $T$  is complete (i.e. for all cwffs  $F$ , either  $T \models F$  or  $T \models \neg F$ );
- 3)  $T$  has exactly one model;
- 4) there exists a model  $\mathcal{M}$  such that for all cwffs  $F$ ,  $T \models F$  iff  $\mathcal{M} \models F$ . □

### 5.2.3. Theories and their models

Two isomorphic structures (see definition 5.5) do not differ on anything essential.

The following theorem confirms this intuition and shows that FOL does not permit us to distinguish between two *isomorphic structures*.

**THEOREM 5.1.**– (structure discrimination). *Let  $\mathcal{M}_1$  and  $\mathcal{M}_2$  denote two isomorphic structures. Then, for all wffs  $F$  of FOL,*

$$\mathcal{M}_1 \models F \text{ iff } \mathcal{M}_2 \models F.$$

**PROOF.**– (intuition). By structural induction (i.e. for all objects introduced by the inductive definition of wffs, see definition 5.1).  $\square$

It is legitimate to wonder whether the converse is also a theorem.

The *negative* answer to this question is given by the following theorem:

**THEOREM 5.2.**– (non-standard model). *There exists a set of formulas  $S$  of FOL that admits the structures  $\mathcal{M}_1 = \langle \mathbb{N}, \{<^{\mathbb{N}}\} \rangle$  and  $\mathcal{M}_2 = \langle D, \{<^{\mathbb{Q}}\} \rangle$  as models (see the definition below), where  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are not isomorphic (example given by Crossley et al.).*

**PROOF.**– Consider the following set of formulas (we indicate as a comment, i.e. a line preceded by %, the desired interpretation of predicate  $P$ ):

- 1)  $\forall x \neg P(x, x)$  % irreflexive
- 2)  $\forall x \forall y. P(x, y) \Rightarrow \neg P(y, x)$  % asymmetric
- 3)  $\forall x \forall y \forall z. P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$  % transitive
- 4)  $\forall x \forall y. P(x, y) \vee P(y, x) \vee x = y$  % total relation. For the axiomatization of  $=$ , see section 9.1.4.
- 5)  $\exists x \forall y. \neg P(y, x)$  % there is an initial element
- 6)  $\forall x \exists y [P(x, y) \wedge \forall z (\neg (P(x, z) \wedge P(z, y)))]$  % every element has a unique immediate successor
- 7)  $\forall x [\exists y (P(y, x) \Rightarrow \forall z (P(z, x) \Rightarrow P(z, y)))]$  % every element, except for the first, has a unique predecessor

It is simple to check that:

$\mathcal{M}_1 = \langle \mathbb{N}, \{<^{\mathbb{N}}\} \rangle$  (where  $<^{\mathbb{N}}$  denotes the usual order relation in  $\mathbb{N}$ ) is a model of  $S$ .

Now consider the following sets:

$$\begin{aligned} D_1 &= \{0\} \cup \{1 - \frac{1}{n} \mid n \in \mathbb{N}, n \neq 0\} && \% [0, 1[ \\ D_2 &= \{1 + \frac{1}{n} \mid n \in \mathbb{N}, n > 1\} && \% ]1, \frac{3}{2}] \\ D_3 &= \{3 - \frac{1}{n} \mid n \in \mathbb{N}, n \neq 0\} && \% [2, 3[ \\ D &= D_1 \cup D_2 \cup D_3 \end{aligned}$$

and the structure  $\mathcal{M}_2 = \langle D, \{<^{\mathbb{Q}}\} \rangle$  (where  $<^{\mathbb{Q}}$  denotes the usual order relation in  $\mathbb{Q}$ ).

It is simple to verify that this is also a model of  $S$ .

But  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are not isomorphic structures, as can be verified by taking an element of  $D$ , for example,  $\frac{3}{2}$  (or  $\frac{4}{3}$ , or  $\frac{5}{4}$ , etc.), which has an *infinity* of predecessors, which is not the case for any element of  $\mathbb{N}$  (see digression 5.3 and definition 5.5).

$\mathcal{M}_2$  is called a non-standard model because it is not isomorphic to the intended model  $\mathcal{M}_1$ , which is called the standard model.<sup>12</sup> Note that Peano's axioms are categorical in second-order logic (SOL).  $\square$

#### 5.2.3.1. How can we reason in FOL?

Similar to what was done in PL, we can provide a syntactical approach (with a formal system) or a semantical approach (based on interpretations). We shall give priority to the latter.

Actually, the methods will be obtained thanks to *semantic notions*, but similar to what was done, for example, for resolution in PL, the goal is to obtain a *deductive system*.

For the syntactical approach, see remark 5.21.

---

<sup>12</sup> If any two models of a theory are isomorphic, then the theory is categorical.

### 5.3. Semantic tableaux in FOL

It is obtained by reduction to the method of semantic tableaux in PL.

There are two fundamental differences compared to the method studied for PL (these differences are inherent to FOL):

- the quantifiers  $\forall$  and  $\exists$  may occur;
- the method may not halt (FOL is *undecidable*).

The way we proceed is basically *intuitive* and is based on the same ideas as the corresponding formal system.

The Löwenheim–Skolem theorem (theorem 5.4) ensures that it is possible to restrict the search for models of FOL formulas to *denumerable* domains.

From now on, we therefore restrict ourselves to denumerable universes (interpretations and models), which are the only ones that can be handled in computer science.

This enables us to solve the problem of handling quantifiers. On a denumerable universe

$$D = \{a_1, a_2, \dots, a_n, \dots\}$$

$\forall$  and  $\exists$  correspond to:

$$\forall xP(x): P(a_1) \wedge P(a_2) \wedge P(a_3) \wedge \dots$$

$$\exists xP(x): P(a_1) \vee P(a_2) \vee P(a_3) \vee \dots$$

(these are not wffs of FOL, but an informal way of stating things).

There are two important questions that need to be asked:

– when variables are replaced by elements of the domain  $D$ , what we obtain is generally not a wff of FOL (see definition 5.1). However, intuition, together with the habits of mathematical practice, suggest that “this does not matter”. Indeed, in mathematical practice, it is not necessary to declare signatures, and symbols are introduced when they are needed. Furthermore, in general, the domain of discourse (semantical point of view) shall be described using a set of symbols that is disjoint from the one used to write formulas (syntactic point of view). We can therefore assume that, each time we are searching for models of FOL formulas (or proving that they do not have any), we have at our disposal a *denumerably infinite* set of *parameters*, denoted by  $Par$  (where  $Par \cap \mathcal{C} = \emptyset$ , see definition 5.1), that will replace the variables occurring in formulas and will also denote the values of closed terms;



– the second key point is that, in the proof of the Löwenheim–Skolem theorem (theorem 5.4), we assume that the considered formula  $F$  is in Skolem normal form (written  $\text{sk}(F)$ ), i.e. in a form that is not necessary to treat the formula with the method of semantic tableaux.

A capital operation to obtain  $\text{sk}(F)$  is the elimination of existential quantifiers by Skolemization. When replacing  $\forall x \exists y P(x, y)$  by  $\forall x P(x, f(x))$ , Skolemization introduces a function  $f$  (which is not defined, but whose existence is guaranteed by the AC). This function can be replaced by its graph with domain and codomain  $Par$ , meaning that  $\forall x \exists y P(x, y)$  will be replaced by  $P(a_1, a_2), P(a_3, a_4), \dots$

Using an argument similar to the one that occurs in the proof of the Löwenheim–Skolem theorem, we could think: if the formula  $\forall x \exists y P(x, y)$  has a model  $\mathcal{M}$  with *uncountably infinite* domain  $D$ , this means (see definition 5.6) that for all  $x \in D$ , there exists  $y \in D$  such that  $P^{\mathcal{M}}(x, y)$  (where  $P^{\mathcal{M}}$  is the relation assigned to  $P$  in the model). But if this is true for *all*  $x \in D$ , it must also hold for a *denumerable* subset of  $D$ , and, in particular, up to a renaming, on the domain  $Par$ ; see also section 5.4.

A similar reasoning applies to formulas that are exclusively quantified with  $\forall$ 's.

We shall use the following equivalent formulas (not doing so can lead to errors; see exercise 5.3):

$$\forall x P(x) \quad \mathcal{K} \quad \forall x Q(x) \text{ is equivalent to } \forall x P(x) \quad \mathcal{K} \quad \forall y Q(y)$$

$$\exists x P(x) \quad \mathcal{K} \quad \exists x Q(x) \text{ is equivalent to } \exists x P(x) \quad \mathcal{K} \quad \exists y Q(y)$$

where  $\mathcal{K} \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ ,

as well as:

$$\neg \forall x \mathcal{P} \text{ is equivalent to } \exists x \neg \mathcal{P}$$

$$\neg \exists x \mathcal{P} \text{ is equivalent to } \forall x \neg \mathcal{P}$$

$$\neg \forall x \neg \mathcal{P} \text{ is equivalent to } \exists x \mathcal{P}$$

$$\neg \exists x \neg \mathcal{P} \text{ is equivalent to } \forall x \mathcal{P}$$

where  $\mathcal{P}$  denotes a wff.

REMARK 5.11.– The last four rules are frequently used. For example, to express that *function  $f$  is not injective*:

$$f \text{ injective: } \forall x \forall y (x \neq y \Rightarrow f(x) \neq f(y))$$

$f$  not injective:  $\neg[\forall x\forall y(x \neq y \Rightarrow f(x) \neq f(y))]$

is equivalent to:

$\exists x\exists y\neg(x \neq y \Rightarrow f(x) \neq f(y))$

which is equivalent to:  $\% (\neg(A \Rightarrow B) \text{ equiv } A \wedge \neg B)$

$\exists x\exists y(x \neq y \wedge \neg(f(x) \neq f(y)))$

which is equivalent to:

$\exists x\exists y(x \neq y \wedge f(x) = f(y))$

□

The rules to use for the method of semantic tableaux in FOL are those of section 3.2, together with the equivalent formulas above and the rules of Figure 5.1.

$\gamma$	$\gamma(t)$	$\delta$	$\delta(t)$
$\forall xF$	$F[x \leftarrow t]$	$\exists xF$	$F[x \leftarrow t]$
$\neg\exists xF$	$\neg F[x \leftarrow t]$	$\neg\forall xF$	$\neg F[x \leftarrow t]$
	$t$ : <i>closed term</i>		$t$ : <b>new</b> <i>constant symbol</i>

**Figure 5.1.** Rules for semantic tableaux (FOL)

REMARK 5.12.– It should be clear that, although they are *syntactically* different from propositions (base formulas), atomic formulas such as  $P(a_1)$  where  $a_1$  is a constant, correspond to the definition of propositions and it is correct to consider them as such. Atomic formulas such as  $P(x)$  are often called *propositional functions*.

To illustrate the ideas, recall  $H(x)$  ( $x$  is a man). It cannot be evaluated as long as the value of  $x$  is not known. This is not the case for  $H(a)$  (Socrates is a man) or  $H(b)$  (The mother of Socrates is a man).

As for PL, a branch will be closed once an *elementary* contradiction has been detected (for example, of the form  $P(a), \neg P(a)$ ), and those are the only ones that can be detected syntactically (i.e. mechanically). □

REMARK 5.13.– The procedure semantic tableaux (FOL) is *non-deterministic*, in other words, choices are free. But when we fail to close a tree, we may wonder if this is due to the fact that it is indeed impossible to close the tree or if we simply chose a bad strategy that indefinitely delayed some choices that would have permitted us to close the tree. Such a strategy is *unfair*.

```

procedure SEMANTIC TABLEAUX (FOL)
% constructs the tree
input: a finite set of cwffs of FOL  $\mathcal{F} = \{f_1, \dots, f_n\}$ 
output: (if the procedure halts) models of  $\mathcal{F}$  or  $\mathcal{F}$  contradictory
% it may not halt (FOL is undecidable)
begin
root of the tree  $\leftarrow \mathcal{F}$ 
while  $\mathcal{F} \neq \emptyset$  and (there remain open branches)
do
if a branch  $\mathcal{B}$  contains two complementary literals, close  $\mathcal{B}$ 
(i.e. put a  $\times$ )
% the corresponding model is not viable
choose  $f_i \in \mathcal{F}$ 
if  $f_i$  of the form  $[\neg]$  ( $F [ < connective > G ]$ )
% [...] : may not exist
then apply rules  $\alpha$  and  $\beta$ : we obtain  $f_i^j$  ( $1 \leq j \leq 2$ )
Graft  $f_i^j$  ( $1 \leq j \leq 2$ ) in all
open branches going through the node labelled by  $f_i$ 
 $\mathcal{F} \leftarrow \mathcal{F} \setminus \{f_i\}$ 
else if  $f_i$  of the form  $\exists xG$ 
then apply rule  $\delta$  and do  $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\}) \cup G[x \leftarrow t]$ 
% we use existential formulas only once
else if  $f_i$  of the form  $\forall xG$  then apply rule  $\gamma$  and do  $\mathcal{F} \leftarrow \mathcal{F} \cup G[x \leftarrow t]$ 
%  $f_i$  is kept indefinitely
enddo

if closed tree
then return  $\mathcal{F}$  contradictory
else return the sets of literals in the open branches
end

```

Figure 5.2. Procedure semantic tableaux (FOL)

Of course, this does not mean that using a *fair* strategy (i.e. a strategy that never delays a choice indefinitely) can transform an undecidable problem into a decidable one, but it permits us to eliminate some *artificial* non-terminating cases.

The following example is convincing. □

EXAMPLE 5.8.— Assume that we want to show the soundness of the following reasoning ( $a$  denotes a constant):

$$\frac{\forall x \exists y P(x, y)}{\exists y P(a, y)}$$

If we apply a fair strategy, we easily prove that this is a correct reasoning:

$$\begin{array}{l}
 1) \forall x \exists y P(x, y) \\
 2) \neg(\exists y P(a, y)) \\
 \quad \downarrow \\
 3) \forall y \neg P(a, y) \quad (2) \\
 \quad \downarrow \\
 4) \exists y P(a, y) \quad (1) \ x \leftarrow a \\
 \quad \downarrow \\
 5) P(a, b) \quad (4) \ y \leftarrow b \\
 \quad \downarrow \\
 6) \neg P(a, b) \quad (3) \ y \leftarrow b \\
 \quad \times \quad (5) \text{ and } (6)
 \end{array}$$

However, if some choices are delayed indefinitely:

$$\begin{array}{l}
 1) \forall x \exists y P(x, y) \\
 2) \neg(\exists y P(a, y)) \\
 \quad \downarrow \\
 3) \exists y P(a, y) \quad (1) \ x \leftarrow a \\
 \quad \downarrow \\
 4) P(a, b) \quad (3) \ y \leftarrow b \\
 \quad \downarrow \\
 5) \exists y P(b, y) \quad (1) \ x \leftarrow b \\
 \quad \downarrow \\
 6) P(b, c) \quad (5) \ y \leftarrow c \\
 \quad \downarrow \\
 7) \exists y P(c, y) \quad (1) \ x \leftarrow c \\
 \quad \downarrow \\
 \quad \vdots
 \end{array}$$

□

EXAMPLE 5.9.– Consider the famous syllogism “Every man is mortal. Socrates is a man. Therefore, Socrates is mortal”. Its translation in FOL is:

$$\begin{array}{l}
 1) \forall x(H(x) \Rightarrow M(x)) \\
 2) \quad \quad H(a) \\
 \hline
 3) \quad \quad M(a)
 \end{array}$$

where:

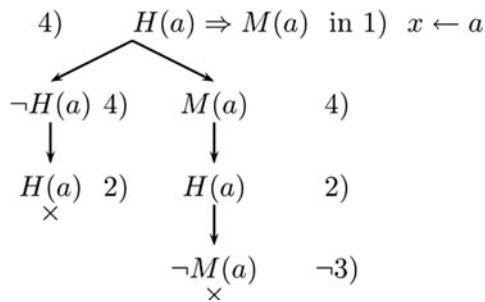
$H(x)$ :  $x$  is a man.

$M(x)$ :  $x$  is mortal.

$a$ : Socrates.

To prove that the reasoning is correct, we consider the set

$$\{\forall x(H(x) \Rightarrow M(x)), H(a), \neg M(a)\}$$

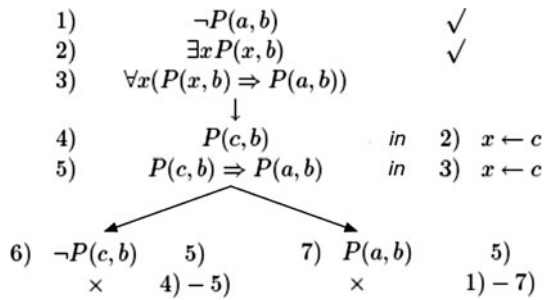


□

EXAMPLE 5.10.– Consider the reasoning:

$$\frac{\forall x(P(x, b) \Rightarrow P(a, b)) \quad \neg P(a, b)}{\neg \exists x P(x, b)}$$

We try to construct models for the set of wffs 1, 2, 3 below (2 is the negation of the conclusion).



Note that, in this example, we used some general principles in the method (indicated in the algorithm) that correspond to normal practices in mathematics:

- we replaced  $\exists x$  by  $c$ , a fresh constant (that did not occur in the considered wffs);
- once we have introduced a constant in place of  $\exists x$ , we are no longer allowed to use (2) (which is why we mark (2) as *used* ( $\checkmark$ ));
- in (3), variable  $x$  can potentially be replaced infinitely many times, which is why it cannot be marked as *used*. We see that  $\forall x$  can cause trees to be *infinite*;

– similar to the case of PL, in which each wff that we used was (implicitly) marked, we mark (1) with  $\checkmark$ .  $\square$

EXAMPLE 5.11.– A problem arises: what happens if the reasoning that we are verifying is not correct, in other words, if the associated set of wffs is not contradictory? The answer is the same as in PL: *there will be open branches*. But the difference is that there may be *infinite* branches in FOL. We examine a case in which the method halts and others in which it does not.

$$\frac{\exists xP(x)}{\forall xP(x)}$$

We thus consider the set of wffs 1,2 below

$$\begin{array}{l} 1) \quad \exists xP(x) \quad \checkmark \\ 2) \quad \neg(\forall xP(x)) \\ \quad \downarrow \\ 3) \quad \exists y\neg P(y) \quad \checkmark \\ \quad \downarrow \\ \quad \neg P(a) \quad \text{in (3)} \quad y \leftarrow a \\ \quad \downarrow \\ \quad P(b) \quad \text{in (1)} \quad x \leftarrow b \end{array}$$

We halt without being able to close the tree. We have a model of the set of wffs, i.e. a counter example of the initial reasoning. The meaning of the open branch is simple: it is a unary relation (i.e. a property) corresponding to  $P$  that is true for  $b$  (i.e.  $b$  belongs to the relation) and false for  $a$  (i.e.  $a$  does not belong to the relation). This interpretation makes the premises true and the conclusion false.

More formally, an interpretation  $\mathcal{I}$  can be extracted from the open branch of the tableaux (see definition 5.6), which is a counter example of the proposed reasoning:

$$\mathcal{M} = \langle D; \{\mathcal{R}\} \rangle$$

with:

$$D = \{a, b\}$$

$$\mathcal{R} = \{b\}$$

in  $\mathcal{I}$ ,  $a_{fp} : P \mapsto \mathcal{R}$

In other cases, it is not possible (for the *method*, a human would easily realize that it will not halt).

If we want to test the validity of the wff:

$$\exists x \forall y \neg P(x, y)$$

we test the unsatisfiability of:

$$\begin{array}{c} \neg(\exists x \forall y \neg P(x, y)) \\ \downarrow \\ \forall x \exists y P(x, y) \\ \downarrow \\ P(a_1, b_1) \\ \downarrow \\ P(a_2, b_2) \\ \downarrow \\ P(a_3, b_3) \\ \vdots \end{array}$$

“we can see” that the method is trying to construct an *infinite* model (think of  $D: \mathbb{N}$  and  $P(x, y): x < y$ ). □

EXAMPLE 5.12.– We want to test whether the following reasoning is correct using the method of semantic tableaux.

$$\frac{\begin{array}{l} \text{All engineers have a university diploma} \\ \text{Some people who have a university diploma are poor} \end{array}}{\text{Some engineers are poor.}}$$

This is formalized in FOL using the following predicates:

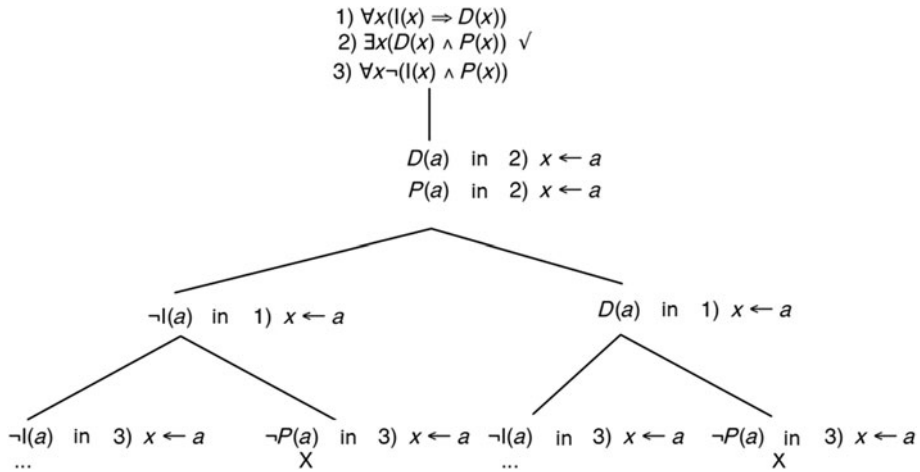
$I(x)$  :  $x$  is an engineer;

$D(x)$  :  $x$  has a university diploma;

$P(x)$  :  $x$  is poor.

$$\frac{\begin{array}{l} \forall x(I(x) \Rightarrow D(x)) \\ \exists x(D(x) \wedge P(x)) \end{array}}{\exists x(I(x) \wedge P(x))}$$

We thus try to construct models of the premises and of the negation of the conclusion.



The leftmost branch cannot be closed (the method does not detect this property, see also theorem 5.11) and provides a counter example of the initial reasoning: someone who has a university diploma ( $D(a)$ ), who is poor ( $P(a)$ ), and who is not an engineer ( $\neg I(a)$ ). We could imagine a universe containing only a poor lawyer, or a universe in which none of those who are poor and have a university diploma are engineers.

Of course, there can be poor engineers, but the proposed reasoning does not prove this. □

EXAMPLE 5.13.— A seldom explored characteristic of semantic tableaux is the possibility to detect in some cases the  $n$ -validity of a formula (which means that a formula is valid in every universe  $D$  such that  $\text{card}(D) \leq n, n \in \mathbb{N}$ ) but can be falsified in any universe of a greater cardinality. We use the method of semantic tableaux to show that the wff

$$\mathcal{A} \Rightarrow \forall x P(x)$$

where:

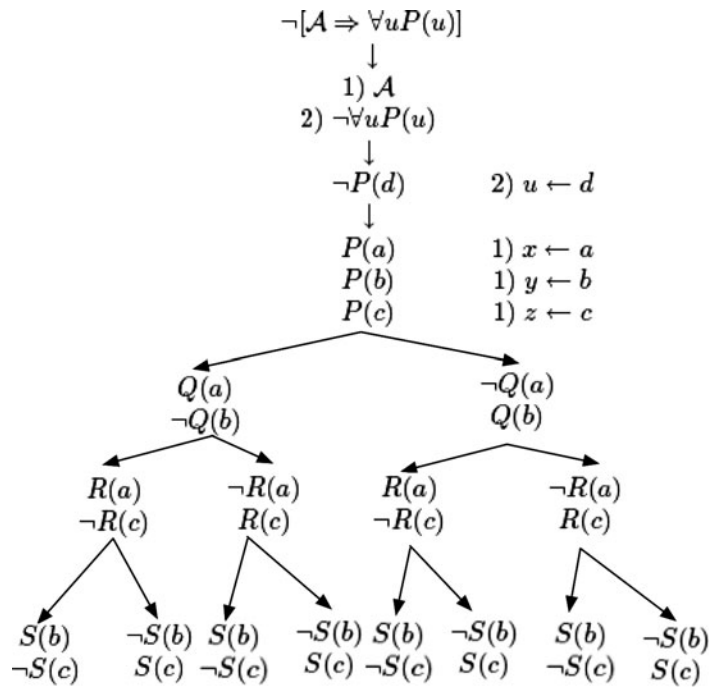
$$\begin{aligned}
 \mathcal{A} : & \ [ \exists x \exists y \exists z ( (P(x) \wedge P(y) \wedge P(z)) \\
 & \quad \wedge (Q(x) \Leftrightarrow \neg Q(y)) \\
 & \quad \wedge (R(x) \Leftrightarrow \neg R(z)) \\
 & \quad \wedge (S(y) \Leftrightarrow \neg S(z)) ) ]
 \end{aligned}$$

is three-valid.

$\mathcal{A}$  states (see *Leibniz's law*, section 9.1.4) that there exist three distinct objects  $x, y, z$ .

The last three conjuncts express the fact that each of the tree objects is different from the other two, as a given object cannot have and not have a property.





We analyze what happens in universes of cardinality 1, 2, 3 (branches are identified with the standard notation on trees):

**card(D) = 1**

$a = b = c = d$  contradiction in branch 1.

**card(D) = 2**

$a = b = c$  contradiction in branches 1.1 and 1.2.

$a = b = d$  contradiction in branch 1.

$a = c = d$  contradiction in branch 1.

$b = c = d$  contradiction in branch 1.

$a = b$  and  $c = d$  contradiction in branch 1.

$a = c$  and  $b = d$  contradiction in branch 1.

$a = d$  and  $b = c$  contradiction in branch 1.

$\text{card}(\mathbf{D}) = 3$

$a = b$  contradiction in branches 1.1 and 1.2.

$a = c$  contradiction in branches 1.1.1; 1.1.2; 1.2.1; and 1.2.2.

$a = d$  contradiction in branch 1.

$b = c$  contradiction in branches 1.1.1.1; 1.1.1.2; 1.1.2.1; 1.1.2.2; 1.2.1.1; 1.2.1.2; 1.2.2.1; and 1.2.2.2.

$b = d$  contradiction in branch 1.

$c = d$  contradiction in branch 1.

For universes  $D$  with  $\text{card}(D) \geq 4$ , no branch can be closed (the open branches provide counter models of the initial formula).  $\square$

REMARK 5.14.– In this example, we implicitly violated the requirements of rule  $\delta$  (i.e. that every constant introduced by an existential quantifier must be a fresh constant that is not introduced by any other existential quantifier).

This violation allows us to *enumerate the models of all cardinalities for which a non-valid formula can be satisfied*.

It is not difficult to show that we have extended the method without losing its soundness and completeness properties.  $\square$

We may wonder whether this example can be generalized to any arbitrary  $n$ . This is the topic of the following exercise.

EXERCISE 5.2.– Can we give a wff in FOL that specifies the  $n$ -validity for  $n \in \mathbb{N}$  (i.e. for an *arbitrary* and *fixed* value of  $n$ )? (See also example 9.31.)  $\square$

EXERCISE 5.3.– Can you prove that following reasonings (respectively, wffs) (a) to (l) given below are correct using the method of semantic tableaux?

a)  $\forall x(P(x) \wedge Q(x)) \Leftrightarrow \forall xP(x) \wedge \forall xQ(x)$

b)  $\exists x(P(x) \vee Q(x)) \Leftrightarrow \exists xP(x) \vee \exists xQ(x)$

c)

$$\frac{\forall x\exists yP(y, x) \quad \forall x\forall y(P(x, y) \Rightarrow Q(x, y))}{\forall x\exists yQ(y, x)}$$

d) Prove that: *every irreflexive<sup>13</sup> and transitive relation is asymmetrical.*

$$\frac{\forall x \neg P(x, x) \quad \forall x \forall y \forall z (P(x, y) \wedge P(y, z) \Rightarrow P(x, z))}{\forall x \forall y (P(x, y) \Rightarrow \neg P(y, x))}$$

e)  $\forall x (P(x) \vee Q(x)) \Rightarrow \forall x P(x) \vee \forall x Q(x)$

f)

$$\frac{\forall x (P(x) \Rightarrow Q(x))}{\forall x (\exists y (P(y) \wedge R(x, y)) \Rightarrow \exists y (Q(y) \wedge R(x, y)))}$$

g)

$$\frac{\forall x (F(x) \Rightarrow \exists y G(x, y)) \quad \exists x F(x)}{\forall x \exists y G(x, y)}$$

h)

$$(\forall x \exists y (P(x) \vee Q(y))) \Leftrightarrow (\forall x P(x) \vee \exists y Q(y))$$

i)

$$(\forall x (P(x) \vee Q(f(x)))) \Rightarrow ((\forall x \exists y (P(x) \vee Q(y)))$$

j) Add a premise to the reasoning of example 5.12 so as to make it correct.

Give the corresponding closed tableaux.

k) Use the method of semantic tableaux to determine if the following formula:

k1) is not valid, and if this is the case, extract a counter example from the tableaux;

k2) is valid, and in this case give a model obtained using the method of semantic tableaux.

$$[\exists x (P(x) \Rightarrow Q(x))] \Rightarrow [\forall x P(x) \Rightarrow \exists x Q(x)]$$

l)  $a, b, c, d, e$  denote constants.

- 1)  $P(a, b)$
- 2)  $P(b, c)$
- 3)  $P(c, d)$
- 4)  $P(d, e)$

---

<sup>13</sup> Irreflexive ( $\forall x \neg P(x, x)$ )  $\neq$  non-reflexive ( $\exists x \neg P(x, x)$ ).

$$5) \forall x \forall y \forall z. P(x, y) \wedge P(y, z) \Rightarrow P(x, z)$$

---


$$6) P(a, e)$$

□

EXERCISE 5.4.– Use the method of semantic tableaux to tell whether the following reasoning is correct or not.

If it is incorrect, extract a counter example of *minimal cardinality* from the tableaux.

1)  $\exists x(P(x) \wedge Q(x))$

2)  $\exists x(R(x) \wedge S(x))$

3)  $\exists x(\neg P(x) \wedge \neg R(x))$

---

 4)  $\exists x(S(x) \wedge Q(x))$

□

#### 5.4. Unification in the method of semantic tableaux

In the method of semantic tableaux for FOL, there are two main problems, the first with rule  $\gamma$ , which corresponds to the instantiation of universally quantified variables, and the other with rule  $\delta$ , which corresponds to the introduction of *fresh* and *unique* constants as instantiations of existentially quantified variables.

– The problem with rule  $\gamma$ , which may generate infinite branches, is to find adequate instances so as to close the branches (that can be closed) at the least possible depth.

A solution that is frequently adopted in implementations is to replace a universally quantified variable, say  $x$ , by a (free) variable  $X$ .

This renaming shall be written as  $x \longrightarrow X$ .

Of course, the “disappearance” of the universal quantifiers must not hide the fact that the free variables that were introduced can be replaced by any number of terms. To keep this property, we introduce renamings of the free variables that we shall note:  $x \longrightarrow X_1, X_1 \longrightarrow X_2, X_2 \longrightarrow X_3, \dots$

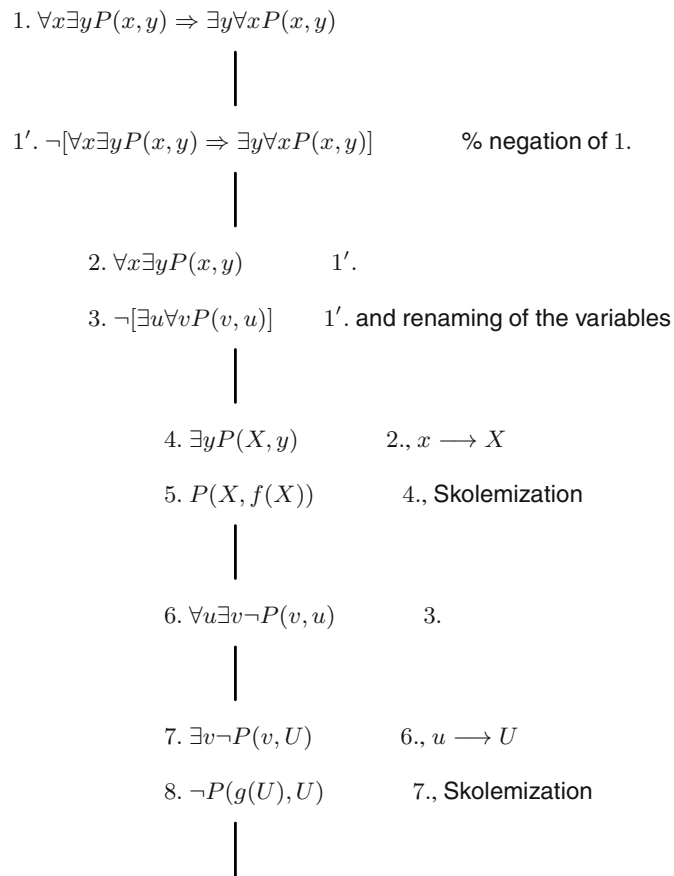
– The problem with rule  $\delta$  is that, when there will be, say,  $n$  free variables  $X_1, X_2, \dots, X_n$  that have been introduced in the branch corresponding to formula  $F$ , where  $\exists y$  appears in the scope of  $X_1, \dots, X_n$ , as usual,  $\exists y$  will be erased, and  $y$  will be replaced by  $f(X_1, X_2, \dots, X_n)$ , where  $f$  is a *new* functional symbol called a *Skolem function*. This guarantees the introduction of a new name (constant) each time there is an instantiation of an existentially quantified variable, which is in the scope of universally quantified variables (syntactically different terms correspond to *different names*), i.e. the most general case.

(This is what we have done systematically, for example, in example 5.11)

– The unification algorithm is used to find the instantiation that allows us to close branches or to detect whether they (still) cannot be closed, therefore, suggesting



EXAMPLE 5.15.– (unification in semantic tableaux, 2). Prove the validity (or non-validity) of the formula:



are arguments (5.–8. ) unifiable?

**Unification fails (cycle)**

**% By renaming  $X \rightarrow X_1, U \rightarrow U_1$  the unification algorithm will always fail (cycle)**

⋮

therefore, the formula is not valid.

□

EXERCISE 5.5.– Show that the following reasoning is correct using the method of semantic tableaux with free variables (i.e. using unification).

$$\forall x \exists y (P(x) \Rightarrow Q(y))$$

---


$$\exists y \forall x (P(x) \Rightarrow Q(y)) \quad \square$$

### 5.5. Toward a semi-decision procedure for FOL

It is *impossible* to design a decision procedure for FOL. Indeed, FOL formulas can be used to describe any Turing machine, and deciding the validity of an arbitrary wff is equivalent to solving the halting problem (which is *undecidable*).

But there is nothing stopping us from searching for a semi-decision procedure for this logic (only hope for the automation of such a procedure)<sup>14</sup>, i.e. a procedure that, if it stops, gives a correct answer, but for which we cannot say anything if it has not stopped at a given moment.

To test if a formula is satisfiable or valid, we (potentially) have to test infinitely many universes, which is impossible. We will see that it will suffice to restrict ourselves to a privileged universe that will have “good properties”. The first step in this direction is to transform the formulas into a normal form.

#### 5.5.1. Prenex normal form

When we consider a set of objects that can have many different forms, it is desirable to have a transformation into a normal form. It permits us, for example, to study the properties of these objects in a uniform manner and to state these properties more easily.

DEFINITION 5.8.– A wff of FOL  $F$  is in prenex normal form, (denoted by  $pr(\mathbf{F})$ ) iff it is of the form:

$$Q_1 x_1 \dots Q_n x_n \mathcal{M} \quad (n \geq 0) \quad (n = 0 \text{ means that the formula contains no quantifier})$$

where  $Q_i$ :  $\forall$  or  $\exists$

$$Q_1 x_1 \dots Q_n x_n \text{ is called the prefix.}$$

---

<sup>14</sup> Another possibility of automation is the use of *heuristics*, but in this case, we cannot guarantee that the procedure will halt with a correct answer in all the cases in which the formula has the expected property.

$\mathcal{M}$ : does not contain any quantifier and is called the matrix.

We implicitly assume that (see definition 5.1) if  $Q_i \neq Q_j$ , then  $x_i \neq x_j$  (meaning that a same variable is not quantified by different quantifiers).

**THEOREM 5.3.**– (existence of the prenex normal form). *If  $F$  is a wff of FOL, then there exists an equivalent  $pr(F)$ .*

**PROOF.**– (outline). Apply the equivalent transformations below and reason by induction on the number of connectives.

The rules assume that  $x \notin Var(G)$  (rename if necessary)

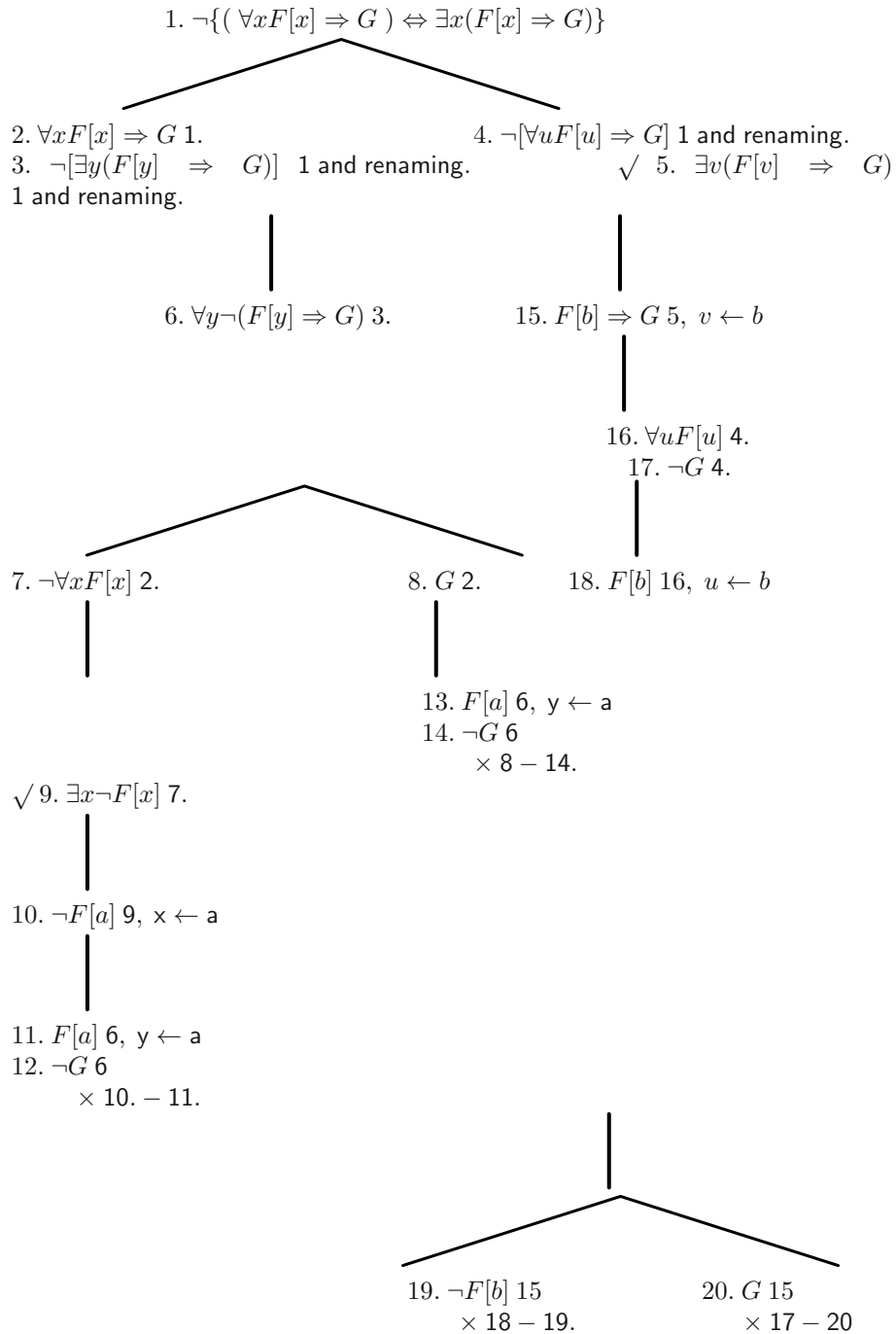
- 1)  $\neg\forall xF$  if and only if  $\exists x\neg F$
- 2)  $\neg\exists xF$  if and only if  $\forall x\neg F$
- 3)  $\forall xF \wedge G$  if and only if  $\forall x(F \wedge G)$
- 3')  $G \wedge \forall xF$  if and only if  $\forall x(G \wedge F)$
- 4)  $\exists xF \wedge G$  if and only if  $\exists x(F \wedge G)$
- 4')  $G \wedge \exists xF$  if and only if  $\exists x(G \wedge F)$
- 5)  $\forall xF \Rightarrow G$  if and only if  $\exists x(F \Rightarrow G)$
- 5')  $G \Rightarrow \forall xF$  if and only if  $\forall x(G \Rightarrow F)$
- 6)  $\exists xF \Rightarrow G$  if and only if  $\forall x(F \Rightarrow G)$
- 6')  $G \Rightarrow \exists xF$  if and only if  $\exists x(G \Rightarrow F)$
- 7)  $\forall xF \vee G$  if and only if  $\forall x(F \vee G)$
- 7')  $G \vee \forall xF$  if and only if  $\forall x(G \vee F)$
- 8)  $\exists xF \vee G$  if and only if  $\exists x(F \vee G)$
- 8')  $G \vee \exists xF$  if and only if  $\exists x(G \vee F)$

1, 2, 3, 4 are sufficient (the other rules can be applied using the equivalent formulas of exercise 3.2).  $\square$

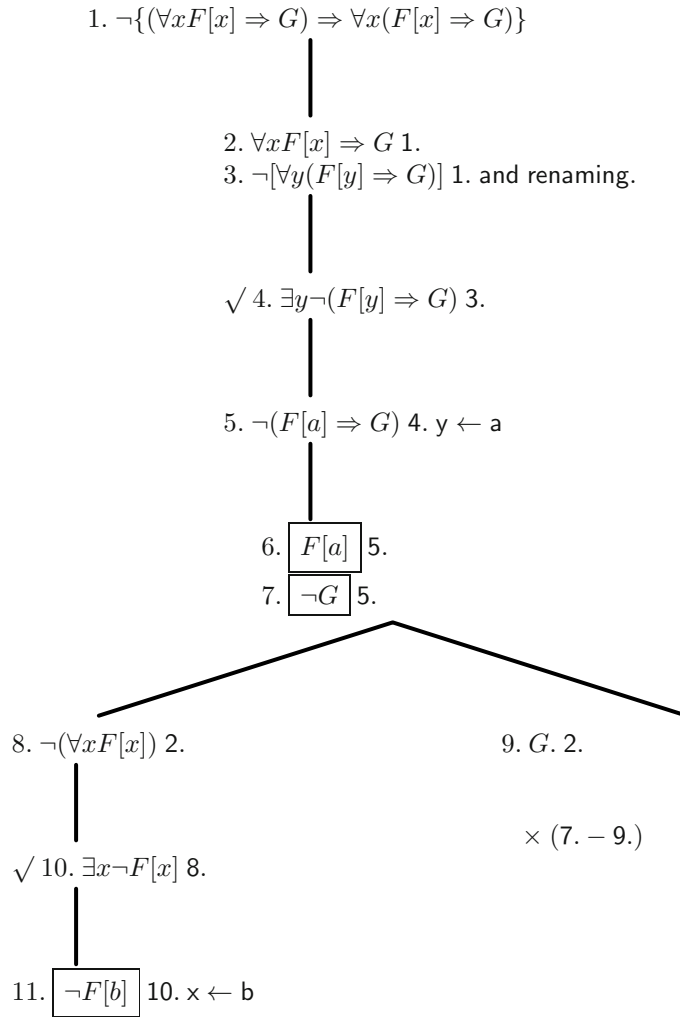
**EXAMPLE 5.16.**– We can use (this is what we will do in exercise 5.7) the method of semantic tableaux to prove the validity or non-validity via a counter example of expressions denoting wffs that we will call “generic formulas” i.e. sets of formulas with a given structure, that can be characterized by naming their subformulas.

We prove below the validity of rule 5, which is used in the proof of theorem 5.3.





We will also use the information provided by the method of semantic tableaux to construct a counter example of an incorrect transformation rule that we might be tempted to use: *as subformula  $G$  does not contain the quantified variable  $x$ , we can include  $G$  in the scope of  $\forall x$ .*



We can extract a model from the negation of the formula (i.e. a counter example of the considered formula) in the open branch (there are no longer any universally quantified formulas: 2 was replaced by 8 and 9), by taking, for example:

$$F[x] \leftarrow P(x)$$

$$G \leftarrow Q(y)$$

(This replacement respects the initial hypotheses.)

and the structure  $\mathcal{M} = \langle D; \mathcal{R} \rangle$

with:

$$D = \{a, b\}$$

and:

$$\mathcal{R} = \{\{a\}, \{b\}\}$$

The counter example constructed by the tree corresponds to:

$$P \mapsto \{a\} \quad \%, \text{ i.e. } P(a): \mathbf{T}; P(b): \mathbf{F}$$

$$Q \mapsto \{b\} \quad \%, \text{ i.e. } Q(b): \mathbf{T}; Q(a): \mathbf{F}$$

This is indeed a counter example, the verification is immediate:

$$(*) (\forall x P(x) \Rightarrow Q(y)) \Rightarrow \forall x (P(x) \Rightarrow Q(y))$$

$$\forall x P(x): \mathbf{F} \quad \%, \text{ as } P(b): \mathbf{F}$$

$$\text{thus, } \forall x (P(x) \Rightarrow Q(y)): \mathbf{T}$$

$$\forall x (P(x) \Rightarrow Q(y)) \mathbf{F} \quad \%, \text{ as } P(a) \Rightarrow Q(a): \mathbf{F}$$

(\*) is therefore evaluated to  $\mathbf{F}$  in the proposed interpretation.  $\square$

Is the prenex normal form *unique*? The following example shows that this is not the case.

EXAMPLE 5.17.— (non-uniqueness of the prenex normal form). The order of application of the rules can lead to different prenex normal forms for the same formula. Consider the formula:

$$\forall x \exists y F \Rightarrow \exists z G \quad (x, y \notin \text{Var}(G), z \notin \text{Var}(F))$$

$$\begin{aligned} & - \forall x \exists y F \Rightarrow \exists z G \xrightarrow{6'} \exists z (\forall x \exists y F \Rightarrow G) \xrightarrow{5} \exists z (\exists x (\exists y F \Rightarrow G)) \xrightarrow{6} \\ & \exists z \exists x \forall y (F \Rightarrow G) \end{aligned}$$

$$\begin{aligned}
& - \forall x \exists y F \Rightarrow \exists z G \xrightarrow{5} \exists x (\exists y F \Rightarrow \exists z G) \xrightarrow{6} \exists x (\forall y (F \Rightarrow \exists z G)) \xrightarrow{6'} \\
& \exists x \forall y \exists z (F \Rightarrow G) \\
& - \forall x \exists y F \Rightarrow \exists z G \xrightarrow{5} \exists x (\exists y F \Rightarrow \exists z G) \xrightarrow{6'} \exists x (\exists z (\exists y F \Rightarrow G)) \xrightarrow{6} \\
& \exists x \exists z \forall y (F \Rightarrow G) \quad \square
\end{aligned}$$

The prefix of the prenex normal form contains existential quantifiers. To establish fundamental theorems for the automation of FOL, we must introduce an operation permitting to eliminate them.

#### 5.5.1.1. Skolemization

*Skolemization* (from the Norwegian logician T. Skolem) enables us to eliminate existential quantifiers while retaining the satisfiability of a formula.

The idea is that:

(\*)  $\forall x \exists y P(x, y)$  admits a model, and  $f$  is a function symbol *that does not appear in the considered formula*

iff:

(\*\*)  $\forall x P(x, f(x))$  admits a model.

but (\*) and (\*\*) are not equivalent, i.e. are not evaluated to the same truth value for the *same* interpretations.

More generally:

the formula:

$$\forall x_1 \forall x_2 \dots \forall x_m \exists \mathbf{y} \forall x_{m+1} \dots \forall x_{m+p} P(x_1, x_2, \dots, x_m, \mathbf{y}, x_{m+1}, \dots, x_{m+p})$$

yields by Skolemization:

$$\begin{aligned}
& \forall x_1 \forall x_2 \dots \forall x_m \forall x_{m+1} \dots \forall x_{m+p} \\
& P(x_1, x_2, \dots, x_m, \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m), x_{m+1}, \dots, x_{m+p})
\end{aligned}$$

and if  $m = 0$ :

$$\forall x_{m+1} \dots \forall x_{m+p} P(\mathbf{a}, x_{m+1}, \dots, x_{m+p})$$

We will show, using the method of semantic tableaux, that:

$$(**) \forall x \exists y P(x, y) \not\equiv \forall x P(x, f(x))$$

meaning that there exist models of  $\forall x \exists y P(x, y)$  that are counter models of  $\forall x P(x, f(x))$ .

We will therefore try to construct models of:

$$S = \{\forall x \exists y P(x, y), \neg[\forall x P(x, f(x))]\}$$

and if we succeed, we will have proved (\*\*).

$$\begin{array}{l} 1) \forall x \exists y P(x, y) \\ 2) \exists z \neg P(z, f(z)) \\ \quad \downarrow \\ \neg P(a, f(a)) \quad 2), z \leftarrow a \\ \quad \downarrow \\ P(a, b) \quad 1), y \leftarrow b, x \leftarrow a \\ \quad \vdots \end{array}$$

Using this tableaux, it is simple to construct the desired counter example (by letting  $f(a) \neq b$ ):

$$\mathcal{M} = \langle D; \{f^{\mathcal{M}}\}, \{P^{\mathcal{M}}\} \rangle$$

with:

$$D = \{a, b\}$$

$$P \mapsto P^{\mathcal{M}} = \{(a, b), (b, a)\}$$

$$f \mapsto f^{\mathcal{M}} = \{(a, a), (b, b)\} \text{ \% total function}$$

Note that the tableaux guarantees that we can always (if  $(a, b) \in P^{\mathcal{M}}$ ) transform a model of  $\forall x \exists y P(x, y)$  into a model of  $\forall x P(x, f(x))$  (by letting  $f(a) = b$ ).

This non-equivalence (sometimes called *weak equivalence*) is not really a problem when we try to prove that a wff *does not have any model* (i.e. that its negation is valid).

If we wanted equivalence to be preserved, we would have to replace:

$$\forall x \exists y P(x, y)$$

by:

$\exists f \forall x P(x, f(x))$ , but this formula *is not a wff of FOL* (see the syntax of FOL in definition 5.1 and exercise 9.5).

REMARK 5.15.— Another way to prove that Skolemization does not preserve equivalence (but only satisfiability) is to consider the following counter example.

The formula:

$$(*) \neg \forall x P(x) \vee P(a)$$

is *valid* (as can easily be checked using the method of semantic tableaux or by noticing that this formula is equivalent to  $\forall x P(x) \Rightarrow P(a)$ ).

If we transform  $(*)$ , we first obtain:

$$\exists x \neg P(x) \vee P(a)$$

and using Skolemization:

$$\neg P(b) \vee P(a)$$

which is a *satisfiable* but *not valid* formula (as  $P(a)$  can be evaluated to **F** and  $P(b)$  to **T**).  $\square$

### 5.5.2. Skolem normal form

Starting with a FOL formula  $F$ , after transformation into prenex normal form and Skolemization, we obtain the *Skolem normal form* of  $F$  (denoted by  $\text{sk}(F)$ ):

$$\text{sk}(F): \forall x_1 \dots \forall x_n F', \text{ where } F' \text{ does not contain any quantifier and } \text{Var}(F') = \{x_1 \dots x_n\}$$

Therefore, by theorem 5.3 and the Skolemization property:

$$(*) F \text{ wff of FOL is satisfiable iff } \text{sk}(F) \text{ is satisfiable}$$

DEFINITION 5.9.— (universe, base, Herbrand interpretation). *Given a finite set  $S$  of formulas in Skolem normal form constructed on the set of predicate symbols  $\pi$  and function symbols  $\mathcal{F}$ , the Herbrand universe on  $\mathcal{F}$  (or Herbrand universe of  $S$ ) is defined as follows:*

$$- H_0 = \{a \mid a \in \mathcal{F}\};$$

*if  $\mathcal{F}$  does not contain any constant, then  $H_0 = \{a\}$ ;*

$$- H_{i+1} = \{f_j^{(n)}(t_1, \dots, t_n) \mid f_j^{(n)} \in \mathcal{F}; \quad t_k \in H_i\} \cup H_i \quad (1 \leq k \leq n; \quad i \geq 0; \quad n \geq 1);$$

$$- H(S) = H_\infty = H_{i \in \mathbb{N}}.$$

(With the notation of definition 4.1, we should note  $\Sigma(\mathcal{F})$  instead of  $H(S)$ .)

The terms in  $H(S)$  are obviously closed (see definition 4.1) and are called Herbrand terms.

The Herbrand base of  $S$  is the set of positive literals (i.e. of atomic formulas):

$$- B(S) = \{L[\bar{x} \mid \bar{t}_{H(S)}] \mid L(\bar{x}) \in C \in S \text{ or } L(\bar{x})^c \in C \in S; \bar{t}_{H(S)} \in H(S)\} \cup \{AF_c\}$$

where  $L[\bar{x} \mid \bar{t}_{H(S)}]$  means all variables in  $L$  are replaced by Herbrand terms.  $L[\bar{x} \mid \bar{t}_{H(S)}]$  is a Herbrand instance or simply constant instance or closed instance of  $L(\bar{x})$ . We shall also talk of (see definition 5.10) closed clauses.

$AF_c$ : closed atomic formulas in  $S$ .

Notation: here, as is customary,  $\bar{x}$  denotes an  $n$ -tuple of variables and  $\bar{t}_{H(S)}$  denotes an  $n$ -tuple of Herbrand terms.

- A Herbrand interpretation of a set of universally quantified formulas  $S$  constructed on  $(\pi, \mathcal{F})$  is an interpretation  $\mathcal{I}_{H(S)}$  such that:

$$D = H(S)$$

$$\text{if } a \in \mathcal{F}: a_{fp}(a) = a$$

$$\text{if } f_j^{(n)} \in \mathcal{F}: a_{fp}(f_j^{(n)}) = f_j^{(n)H(S)}$$

with:

$$f_j^{(n)H(S)} : (t_1, \dots, t_n) \mapsto f_j^{(n)H(S)}(t_1, \dots, t_n)$$

(and of course  $t_i \in H(S); 1 \leq i \leq n$ )

$$\text{if } P_k^{(m)} \in \pi: a_{fp}(P_k^{(m)}) = P_k^{(m)H(S)}$$

with  $P_k^{(m)H(S)} \subseteq H(S)^m$

- A Herbrand interpretation can be represented by  $\mathcal{I}_{H(S)} \subseteq B(S)$ .

The obvious intuitive meaning is that the literals  $L^n(\bar{t}) \in \mathcal{I}_{H(S)}$  are evaluated to  $\mathbf{T}$  on the  $n$ -tuples  $\bar{t} \in H(S)^n$  or that  $\bar{t} \in L^{(n)H(S)}(\bar{t})$ .

The set of Herbrand interpretations of  $S$  is thus the set of subsets of  $B(S)$ .

REMARK 5.16.– The Herbrand universe corresponds to what is called the term algebra in universal algebras., i.e. (see definition 5.4):

$\langle \Sigma(\mathcal{F}), \mathfrak{F} \rangle$  (with  $\Sigma(\mathcal{F}) \neq \emptyset$ ) (condition that is satisfied if  $\mathcal{F} \neq \emptyset$ ) and

$$f^{(n)\Sigma(\mathcal{F})} : (t_1, \dots, t_n) \mapsto f^{(n)}(t_1, \dots, t_n)$$

$$f^{(n)\Sigma(\mathcal{F})} \in \mathfrak{F}$$

$$f^{(n)} \in \mathcal{F}; t_i \in \Sigma(\mathcal{F}) \quad (1 \leq i \leq n)$$

□

EXAMPLE 5.18.–  $S = \{C_1, C_2, C_3\}$

$$C_1 : \forall x P(x)$$

$$C_2 : \forall x. \neg P(x) \vee Q(f(x))$$

$$C_3 : \neg Q(f(a))$$

$$H(S) = \{a, f(a), \dots, f^n(a), \dots\}$$

$$B(S) = \{P(a), Q(f(a)), P(f(a)), Q(f(f(a))), \dots\}$$

a Herbrand interpretation:

$$\mathcal{I}_{H(S)} = \{P(a), Q(f(a))\}$$

meaning that  $P(a)$  is evaluated to **T** and that  $P$  is evaluated to **F** on all other elements of  $H(S)$ .  $Q(f(a))$  is evaluated to **T** and  $Q$  is evaluated to **F** on all other elements of  $H(S)$ .

Another Herbrand interpretation:

$$\mathcal{I}_{H(S)} = \{P(a), P(f(a)), P(f(f(a))), \dots\}$$

meaning that  $P$  is evaluated to **T** on all elements of  $H(S)$ , and  $Q$  to **F** on all elements of  $H(S)$ . □

EXERCISE 5.6.– Let  $S$  denote the following set of wffs:

$$S = \{P(a), \forall x. P(f(x)) \vee \neg Q(f(x)), Q(a)\}$$

Can you give three Herbrand models of  $S$ ? Which ones? □



REMARK 5.17.– By construction, the Herbrand universe (and the Herbrand base) of a set of formulas  $S$  is either *finite* (if no function symbol occurs in  $S$ ) or *denumerably infinite*.

As a consequence, the number of Herbrand interpretations for a set of formulas  $S$  is either *finite* or *uncountably infinite* (see exercise 3.1).  $\square$

For example, we can give an *uncountably infinite* model of the formula  $\forall x \exists y P(x, y)$ , for example,  $\langle \mathbb{R}, \langle \rangle \rangle$ , and we can also give a *denumerably infinite* one, for example,  $\langle \mathbb{N}, \langle \rangle \rangle$ . This example is a particular instance of the following theorem.

THEOREM 5.4.– (Löwenheim–Skolem). *If a FOL wff  $F$  has a model, then it also has a denumerable model.*

PROOF.–

- It suffices to consider  $sk(F)$ , see (\*) in section 5.5.2.
- We assume that  $D$  is the domain of discourse of the model of  $sk(F)$ .
- If a universally quantified wff (such as  $sk(F)$ ) is satisfied on a domain  $D$ , then it is also satisfied on a domain  $D' \subseteq D$  (see definition 5.6 and remark 3.39).
- We consider  $H(sk(F))$  from which we will construct a denumerable set  $D'$ .
- We begin by the (necessarily finite) set of constants in  $F$  or by the additional constant, which will denote, in the model, elements of  $D$ . These will also be elements of  $D'$ .
- The terms  $f^{(n) H(sk(F))}(t_1, \dots, t_n)$  denote elements of  $D$  in the model (see definition 5.6) that are added to  $D'$  ( $D'$  is closed by this operation, by definition of a Herbrand universe).
- By construction  $D'$  is either finite or denumerably infinite (see remark 5.17). This proves the theorem.  $\square$

EXAMPLE 5.19.– Consider the wff:

$$F : \forall x \exists y P(x, y)$$

with the intended interpretation: *for all reals there exists a greater real*;

we obtain by Skolemization:

$$sk(F) : \forall x P(x, f(x))$$

A model can be constructed on the structure:

$$\mathcal{M} = \langle \mathbb{R}; \{f^{\mathbb{R}}\}, \{<^{\mathbb{R}}\} \rangle$$

where  $f^{\mathbb{R}} : \mathbb{R} \rightarrow \mathbb{R} ; f^{\mathbb{R}}(x) = x + 0.5$

$$f \mapsto f^{\mathbb{R}} \text{ and } P \mapsto <^{\mathbb{R}}$$

$$H(sk(F)) = \{a, f^n(a) \mid n \geq 1\} = \{f^n(a) \mid n \geq 0\}$$

if, for example, we fix  $a \mapsto -5$ , we obtain

$$f(a) \mapsto -4.5$$

$$f^2(a) \mapsto -4$$

$$f^3(a) \mapsto -3.5$$

$$f^4(a) \mapsto -3$$

$\vdots$

$$D' = \{-5, -4.5, -4, -3.5, -3, \dots\}$$

A Herbrand model:

$$M = B(sk(F)) = \{P(x, f(x)) \mid x \in H(sk(F))\}. \quad \square$$

We have the following immediate corollary:

**COROLLARY 5.1.**– (Löwenheim–Skolem for finite sets of formulas). *If  $S$  is a finite set of FOL wffs that admits a model, then  $S$  admits a denumerable model.*

**PROOF.**– Let  $S = \{f_1, f_2, \dots, f_n\}$ .

By definition (see definition 3.8)  $S$  has a model iff there exists a model that satisfies all the formulas in  $S$ , i.e. iff:

$$F : \bigwedge_{i=1}^n f_i \text{ has a model.}$$

The corollary is proved by applying theorem 5.4 to  $F$ .  $\square$

**REMARK 5.18.**– The Löwenheim–Skolem theorem also applies to denumerably infinite wffs of FOL: *if  $S$  is a denumerably infinite set of wffs of FOL that has a model, then  $S$  has a denumerable model.*  $\square$

REMARK 5.19.– (an important consequence of the Löwenheim–Skolem theorem). An immediate consequence of the Löwenheim–Skolem theorem is that the intended interpretation of a wff of FOL is not unique: there can also be *unintended interpretations*.

As a consequence, FOL formulas cannot characterize uncountably infinite sets<sup>15</sup>, because we know that along with such a set, they will also admit a denumerable model (i.e. they will *also* characterize a denumerable set).  $\square$

The Löwenheim–Skolem theorem permits us to imagine a mechanical *validity* test (as it will be performed on *denumerable* domains).

There remain two questions that, if answered positively, make the existence of such a test impossible:

- would we have to test *all* interpretations on a given denumerable domain?
- would we have to test *all* denumerable domains?

The answer to both questions is (fortunately) negative.

REMARK 5.20.– (on interpretation domains). Before we formalize these answers, note that the *only* requirement on the domain of an interpretation is that it should not be empty. It can, in particular, consist of terms without variables (closed terms) constructed on the same signature as (the Skolem normal form) of the initial formula.  $\square$

The negative answer to the first question is a result of the following theorem.

THEOREM 5.5.– *We can test the validity of a wff of FOL on a denumerable domain without taking interpretations into account.*

PROOF.–

- Let  $F$  denote the wff whose validity we want to test.
- $\neg F$  is satisfiable iff  $sk(\neg F)$  is sat (see section 5.5.2).
- $F$  is valid iff  $\neg F$  is unsatisfiable.
- $\neg F$  is unsatisfiable iff  $sk(\neg F)$  is unsatisfiable.
- To test the validity of  $F$ , we therefore test whether  $sk(\neg F)$  is unsatisfiable.

---

<sup>15</sup> I.e. specify uncountably infinite sets and none other.

– Let  $D$  denote the domain on which we want to test the unsatisfiability of  $sk(\neg F)$ ;  
 $D = \{a_1, a_2, \dots, a_n, \dots\}$ .

– We assume that  $card(Var(sk(\neg F))) = k > 0$  ( $Var$  is the set of variables and all variables are universally quantified by definition of the Skolem normal form).

– We want to enumerate the  $k$ -tuples of elements of  $D$ , that will be denoted by  $\bar{a}_i, i \in \mathbb{N}$ .

% A bijection  $\mathbb{N}^k \rightarrow \mathbb{N}$  can be defined by:

$$(a_1, a_2, \dots, a_k) \mapsto 2^{a_1} + 2^{a_1+a_2+1} + 2^{a_1+a_2+a_3+2} + \dots + 2^{a_1+a_2+\dots+a_k+k-1} - 1$$

– Testing the universally quantified formula  $sk(\neg F)$  reduces to testing  $sk(\neg F)$  on all the elements of  $D^k$ .

– We therefore consider  $\bigcup_{i \in \mathbb{N}} sk(\neg F)[\bar{x} \mid \bar{a}_i]$  %  $\bar{x}$ :  $k$ -tuples of variables of  $sk(\neg F)$ .

– **If**  $\bigcup_{i \in \mathbb{N}} sk(\neg F)[\bar{x} \mid \bar{a}_i]$  is unsatisfiable **then**  $\exists j \in \mathbb{N}$  such that  $\bigcup_{i=1}^j sk(\neg F)[\bar{x} \mid \bar{a}_i]$  is unsatisfiable (propositional test; contrapositive of the compactness theorem for PL, theorem 3.3).

– This test is independent of the interpretation of the function symbols (and, in particular, of the constants) and of the predicates.  $\square$

The negative answer to the second question is given by the following theorem.

**THEOREM 5.6.**– *A wff for FOL  $F$  is satisfiable iff it is satisfiable on the domain  $H(sk(F))$ .*

**PROOF.**– *If:*

– We consider  $sk(F)$ .

–  $F$  is satisfiable iff  $sk(F)$  is satisfiable (see section 5.5.2).

– By hypothesis,  $sk(F)$  is satisfiable on  $H(sk(F))$  % (see remark 5.20).

– Therefore,  $F$  is satisfiable.

*Only*

– If  $F$  is satisfiable, then  $F$  is satisfiable on denumerable domain  $D$  (Löwenheim–Skolem theorem, theorem 5.4).

– We repeat the reasoning of theorem 5.4, but instead of taking elements in  $D$ , we take the Herbrand terms that denote them to construct another denumerable set  $D'$ .

–  $D' = H(sk(F))$ .  $\square$

The following theorem, which simply merges theorems 5.5 and 5.6, is essential for the automation of FOL.

**THEOREM 5.7.– (Herbrand).** *A wff of FOL  $F$  is valid iff there exists a finite set of instances of  $sk(\neg F)$  that is unsatisfiable (contradictory).*

**PROOF.–**

- $F$  is valid iff  $\neg F$  is unsatisfiable.
- $\neg F$  is satisfiable iff  $sk(\neg F)$  is satisfiable (see section 5.5.2).
- $sk(\neg F)$  is unsatisfiable iff  $HI(sk(\neg F))$  is unsatisfiable % HI: Herbrand Instances.

(semantic definition of universally quantified formulas, see definition 5.6)

– If  $HI(sk(\neg F))$  is unsatisfiable then there exists a finite subset of  $HI(sk(\neg F))$  that is unsatisfiable, (up to a syntactic modification, HI are *propositional*, and we may apply the (contrapositive of) the compactness theorem for PL (theorem 3.3)).  $\square$

It is clear that Herbrand’s theorem enables us to design a semi-decision procedure for FOL:

```

procedure SEMI_DEC_PROC_FOL
input:  $sk(\neg F)$   $Var(sk(\neg F)) = \{x_1, \dots, x_n\}$ 
output:  $F$  valid or  $F$  not valid or  $\perp$ 
halting test: detection of a contradiction or test of all HI (if the universe is finite)
begin
  • Generate  $H(sk(\neg F))$  % Herbrand universe
  • Enumerate the  $n$ -tuples of  $H(sk(\neg F))$ :  $\bar{t}_i$  ( $1 \leq i; i \in \mathbb{N}$ )
  • Obtain  $sk(\neg F)[\bar{x} | \bar{t}_i]$  % HI: Herbrand instances
  • Test (propositionally)  $\bigcup_i \{ sk(\neg F)[\bar{x} | \bar{t}_i] \}$  ( $1 \leq i; i \in \mathbb{N}$ )
  • If contradiction detected then return ‘ $F$  valid’
  • Else if tests exhausted ( $H(sk(\neg F))$  finite) then return  $F$  not valid
  • Else continue
end
    
```

**Figure 5.3.** *Semi-decision procedure for FOL*

**REMARK 5.21.– (completeness of FOL).** Gödel’s *completeness theorem*<sup>16</sup> for FOL can be stated as follows: *there exists a formal system in which every valid FOL formula has a proof.*

The following is a formal system for FOL.

---

<sup>16</sup> Should not be confused with Gödel’s *incompleteness theorem*, which is much more popular (see remark 3.28 and section 5.9).

$\mathcal{S}_{FOL} = \langle \mathcal{L}_1, \mathcal{R}_1, \mathcal{A}_1 \rangle$  with:

$\mathcal{L}_1$ : see definition 5.1

$\mathcal{R}_1$ : the inference rule schema  $MP$  from  $\mathcal{S}_1$  together with the inference rule schemas  $G$  and  $P$  below:

$$G : \frac{F \Rightarrow G(x)}{F \Rightarrow \forall x G(x)}$$

$$P : \frac{G(x) \Rightarrow F}{\exists x G(x) \Rightarrow F}$$

$x \notin \text{Var}(F)$

$\mathcal{A}_1$ : the axiom schemas (A1), (A2), and (A3) of  $\mathcal{S}_1$  together with (A4) and (A5) below:

$$(A4) \forall x F(x) \Rightarrow F(t) \quad t: \text{a term}$$

$$(A5) F(t) \Rightarrow \exists x F(x) \quad t: \text{a term}$$

From a *conceptual* point of view, a formal system can be characterized as a “machine that produces (and tests) theorems in a given logic”, we can therefore view SEMI\_DEC\_PROC\_FOL as a formal system for FOL and theorem 5.7 as a *completeness theorem* for FOL.

This theorem enables us to replace a test of *validity* on *all* interpretations (*semantical* notion) by a test of the existence of a proof (*syntactical* notion of a *proof*).  $\square$

REMARK 5.22.– (very, very long proofs). To illustrate the idea, consider  $\mathcal{S}_{FOL}$  from remark 5.21, but the property below holds for all systems  $\mathfrak{S}$  for which  $\vdash_{\mathfrak{S}} T$  ? is undecidable (immediate by analyzing the proof).

Property:

The *length of a theorem*  $T$ , denoted by  $\text{length}(T)$ , is the number of symbols (from the vocabulary of the corresponding formal system) in its statement. Similarly, the *length of a proof* is the number of symbols in the proof (other definitions can be given such as number of steps, etc.).

For any recursive function (i.e. computable by an algorithm)  $f$ , there exist a wff  $T$  of FOL such that  $\vdash_{\mathcal{S}_{FOL}} T$ , with  $\text{length}(T) = n$  ( $n \in \mathbb{N}$ ) such that the shortest proof of  $T$  has at least length  $f(n)$ .

PROOF.–

- Assume that this property does not hold.
- There therefore exists a recursive function  $F$  for which every theorem  $T$  of length  $n$  has a proof of length at most  $F(n)$ .
- It thus suffices to enumerate the finite set of all sequences of length, at most  $F(n)$  constructed on a finite vocabulary to identify a proof of  $T$ .
- This would be a decision procedure enabling us to decide whether a wff is a theorem of  $\mathcal{S}_{L1O}$ . Such a procedure cannot exist and we obtain a contradiction.
- Note that there exist theorems in  $\mathcal{S}_{L1O}$  of length, say,  $N$ , such that their shortest proofs are of length  $10^{100 \times N}$ , for example.  $\square$

It is natural to wonder whether there is a compactness theorem for FOL.

The answer is the topic of the next theorem.

THEOREM 5.8.– (compactness of FOL).

$S$ : set of wffs of FOL.

If every finite subset of  $S$  is satisfiable, then  $S$  is satisfiable.

PROOF.–

- If  $S$  is finite, the proof is trivial, as  $S$  is satisfiable by hypothesis.
- $S = \{F_1, F_2, \dots, F_n, \dots\} = \{F_i \mid i \in \mathbb{N}\}$  (see remark 5.2).
- If a set of formulas is satisfiable, then so are all its formulas (Warning! The converse is not true, as all the formulas in a set have to be true in the *same interpretation* for a set of formulas to be satisfiable).
- Instead of  $S$ , we can consider  $S' = \{sk(F_i) \mid i \in \mathbb{N}\}$  (see section 5.5.2).
- It is sufficient to consider  $H(sk(F_i))$  as a domain to test satisfiability (see theorem 5.6).
- We enumerate the (increasing) finite sets of closed instances (i.e. *propositional*) of:
 
$$S_1 = \{sk(F_1)\}, S_2 = \{sk(F_1), sk(F_2)\}, \dots, S_n = \{sk(F_1), sk(F_2), \dots, sk(F_n)\}, \dots$$
- As each formula involves a finite number of symbols, the sets of propositional formulas  $S_i ; i \in \mathbb{N}$  are finite; thus, by hypothesis (and theorem 5.6),  $S_i ; i \in \mathbb{N}$  is satisfiable.
- By applying the compactness theorem for PL, we conclude that  $\bigcup_{i \in \mathbb{N}} S_i$  is satisfiable.
- By applying theorem 5.6, we conclude that  $S$  is satisfiable.  $\square$

### 5.6. Semantic trees in FOL

A “natural” way of implementing procedure SEMI\_DEC\_PROC\_FOL is to use semantic trees.

REMARK 5.23.– We do not have to restrict ourselves to formulas in clausal form, but the method can be applied to formulas in Skolem normal form, i.e. that are universally quantified (see definition 5.10 and remark 3.31).  $\square$

The method of semantic trees for FOL is *the same* as that for PL, by considering literals on the Herbrand base  $B(S)$ .

Indeed, by considering, for example, the set of formulas from example 5.18 and the universe  $H(S)$ :

$$C_1: \forall x P(x) \text{ equiv. } P(a) \wedge P(f(a)) \wedge P(f(f(f(a)))) \wedge \dots$$

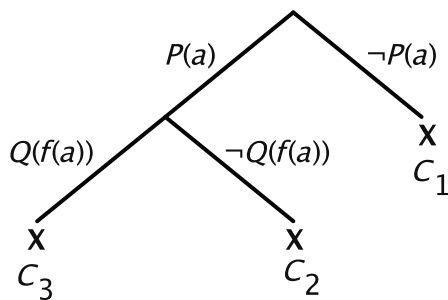
$$C_2: \forall y (\neg P(y) \vee Q(f(y))) \text{ equiv. } (\neg P(a) \vee Q(f(a))) \wedge (\neg P(f(a)) \vee Q(f(f(a)))) \wedge \dots$$

$$C_3: \neg Q(f(a))$$

To produce a counter model of  $S$ , it is sufficient to give a Herbrand interpretation that is a counter model of a closed instance of a formula (i.e. of a conjunct on the right-hand side of “equiv.”).

The following example uses the method of semantic trees to show that the set of formulas  $S$  of example 5.18 is unsatisfiable.

EXAMPLE 5.20.–



where  $C_i$  ( $1 \leq i \leq 3$ ) under  $\times$  means: this interpretation falsifies a closed instance of formula  $C_i$ .



The leftmost branch in the tree corresponds to the first Herbrand interpretation of  $S$  given in example 5.18.

Of course, the tree that is constructed depends on the enumeration order of  $B(S)$ .  $\square$

Just as for resolution in PL, the resolution rule in FOL applies to *sets of clauses*. It is therefore necessary to define clauses in FOL.

DEFINITION 5.10.– (FOL clause, clausal form).

*A clause is a wff of the form:*

$$\forall x_1 \dots \forall x_n \mathcal{P}$$

where:

$$\text{Var}(\mathcal{P}) = \{x_1, \dots, x_n\}$$

(i.e. all the variables in  $\mathcal{P}$  are universally quantified)

$$\mathcal{P} : \bigvee_{i=1}^m L_i$$

*Clauses are usually denoted:*

$$\bigvee_{i=1}^m L_i$$

(variables are implicitly quantified universally)

$L_i$ : literal, i.e. of the form:

$$L_i^p(t_1, \dots, t_p) \text{ or } \neg L_i^p(t_1, \dots, t_p)$$

*A formula is in clausal form iff it is of the form  $\bigwedge_{i=1}^n C_i$ , where the  $C_i$ 's are clauses.*

*More frequently, as it was the case in PL (see remark 3.33), we will mention sets of clauses (or the set of clauses corresponding to a formula)  $S = \{C_1, \dots, C_n\}$ , and we will say that clauses are sets of literals.*

*If  $\mathcal{C}$  and  $\mathcal{D}$  are different clauses, then we always have  $\text{Var}(\mathcal{C}) \cap \text{Var}(\mathcal{D}) = \emptyset$  (possibly after a renaming of the variables).*

The clausal form of a wff  $F$  can be obtained from  $\text{sk}(F)$  and the rules on logical connectives.

**5.6.1. Skolemization and clausal form**

Skolemization can make things complicated. Some preprocessing can sometimes be useful.

$$1) \forall x \exists y (P(y) \vee Q(x))$$

yields by Skolemization:

$$1') \forall x (P(f(x)) \vee Q(x))$$

(1') generates an *infinite* Herbrand universe

but (1) is equivalent to:

$$\exists y P(y) \vee \forall x Q(x)$$

(see exercise 5.3 h)) and rename  $P \leftarrow Q$ ;  $Q \leftarrow P$  )

and after Skolemization we obtain the clause:

$$P(a) \vee \forall x Q(x)$$

which generates a *finite* Herbrand universe;

and sometimes preprocessing makes matters worse.

$$2) \exists x (P(x) \vee Q(x))$$

yields by Skolemization:

$$P(a) \vee Q(a)$$

This clause generates a Herbrand universe containing one constant;

but (2) is equivalent to:

$$\exists x P(x) \vee \exists x Q(x)$$

which, by Skolemization, yields:

$$2') P(a) \vee Q(b)$$

(2') generates a Herbrand universe containing two constants.

Sometimes it can “hide” information:

the formula:

$$\forall xP(x) \vee \neg\forall xP(x)$$

is obviously a tautology. Skolemizing this formula yields:

$$\forall xP(x) \vee \exists x\neg P(x)$$

and we obtain  $\forall xP(x) \vee \neg P(a)$ ,

which is equivalent to:

$$3) \forall x(P(x) \vee \neg P(a))$$

clause (3) will needlessly increase the size of the search space.

It can also increase the size of the problem specification. For the following formula:

$$[\exists x\forall y(P(x) \Leftrightarrow P(y)) \Leftrightarrow (\exists xQ(x) \Leftrightarrow \forall yP(y))] \Leftrightarrow [\exists x\forall y(Q(x) \Leftrightarrow Q(y)) \Leftrightarrow (\exists xP(x) \Leftrightarrow \forall yP(y))]$$

the standard clausal form transformation yields 1,600 clauses!

REMARK 5.24.– The notion of Skolemization is related to the one used in the *proof-as-programs* approach, in which an algorithm is extracted from a constructive proof of a program specification.

The intuitionistic version of Church's thesis<sup>17</sup> goes as follows: if  $Ar$  is a first-order predicate of arithmetic,  $x, y \in \mathbb{N}$  and  $\forall x\exists yAr(x, y)$ , then there exists an algorithm  $f$  such that  $\forall xAr(x, f(x))$ .  $\square$

Herbrand's theorem and the procedure SEMI\_DEC\_PROC\_FOL enable us to obtain the following theorems as immediate corollaries.

THEOREM 5.9.– *A set of clauses  $S$  is unsatisfiable iff for any semantic tree associated to  $S$  there exists a semantic tree that is closed and finite.*

THEOREM 5.10.– (Herbrand's theorem (for clauses)). *A set of clauses  $S$  is unsatisfiable iff there exists a finite set of Herbrand instances of  $S$  that is unsatisfiable.*

---

<sup>17</sup> Its classical version: a function is computable iff it is intuitively computable.

### 5.7. The resolution method in FOL

One may wonder whether the resolution rule also applies in FOL. The answer is yes.

In a famous paper from 1965, J.A. Robinson combined the resolution rule for PL with the unification algorithm, which led to a calculus with a *unique rule* permitting us to automate FOL.

This rule is defined as follows.

DEFINITION 5.11.– (resolution rule for FOL).

$$\frac{P(t_1^1, \dots, t_n^1) \vee \dots \vee P(t_1^p, \dots, t_n^p) \vee \alpha \quad \neg P(s_1^1, \dots, s_n^1) \vee \dots \vee \neg P(s_1^q, \dots, s_n^q) \vee \beta}{\sigma(\alpha \vee \beta)}$$

If  $\sigma$  is the mgu of  $\{t_i^1 \doteq t_i^j \mid j \in [1..p], i \in [1..n]\} \cup \{s_i^1 \doteq s_i^j \mid j \in [1..q], i \in [1..n]\} \cup \{t_i^1 \doteq s_i^1 \mid i \in [1..n]\}$ .

This means that the resolvent clause is obtained by unifying all literals  $P(t_1^i, \dots, t_n^i)$  ( $1 \leq i \leq p$ ), and  $P(s_1^j, \dots, s_n^j)$  ( $1 \leq j \leq q$ ), then applying the rule as in the propositional case.

In practical implementations, the rule above is replaced by *binary resolution* (i.e.  $p = q = 1$ , we select two complementary literals, one in each clause) and a *factorization* rule that aims at unifying two literals with the same predicate symbols and the same sign that occur in a same clause.

DEFINITION 5.12.– (binary resolution rule for FOL). *Given two clauses:*

$$C_1: L(t_1, \dots, t_n) \vee \alpha$$

$$C_2: L^c(t'_1, \dots, t'_n) \vee \beta$$

where:

$\alpha, \beta$ : disjunctions of literals.

If the system of equations:

$$\{t_1 \doteq t'_1, \dots, t_n \doteq t'_n\}$$

has mgu  $\sigma$  as a solution.

The binary resolution rule between  $C_1$  and  $C_2$  is defined by:

$$\mathcal{R}_{b\text{-fol}} : \frac{L(t_1, \dots, t_n) \vee \alpha \quad L^c(t'_1, \dots, t'_n) \vee \beta}{\sigma(\alpha \vee \beta)}$$

Applying a substitution  $\sigma$  to an  $n$ -tuple of terms is defined as follows:

$$\sigma \bar{t} = \sigma(t_1, t_2, \dots, t_n) = (\sigma t_1, \sigma t_2, \dots, \sigma t_n)$$

Applying a substitution  $\sigma$  to a clause is defined as follows:

$$\sigma(L_1(\bar{t}_1) \vee L_2(\bar{t}_2) \vee \dots \vee L_n(\bar{t}_n)) = L_1(\sigma \bar{t}_1) \vee L_2(\sigma \bar{t}_2) \vee \dots \vee L_n(\sigma \bar{t}_n)$$

We will note  $\mathcal{R}_{b\text{-fol}}(C_1, C_2)$  or simply  $C = \mathcal{R}(C_1, C_2)$  (when there is no ambiguity).  $C$  is called the resolvent.  $C_1$  and  $C_2$  the parent clauses

DEFINITION 5.13.– (factorization). The restriction to two complementary literals causes the loss of completeness. To recover completeness, we need to add the so-called factorization rule which, starting with clause:

$$C : P(\bar{t}_1) \vee P(\bar{t}_2) \vee \alpha$$

with  $\alpha$ : disjunction of literals (that may contain other literals of the form  $P(\bar{s})$ ) and  $\bar{t}_1$  and  $\bar{t}_2$  unifiable, generates:

$$D : \sigma[P(\bar{t}_1) \vee \alpha]$$

where  $\sigma$  is the mgu of  $\bar{t}_1$  and  $\bar{t}_2$

$D$  is called factor of  $C$ .

A clause can have several factors.

EXAMPLE 5.21.– (the need for factorization to retain completeness). The set of clauses  $\{P(x) \vee P(y), \neg P(z) \vee \neg P(u)\}$  is unsatisfiable, but binary resolution generates infinitely many clauses of the form

$$P(x) \vee \neg P(y)$$

without generating  $\square$ .

However, the factorization rule generates  $P(x)$  and  $\neg P(z)$ , and the binary resolution rule generates  $\square$ .  $\square$

DEFINITION 5.14.–

– Given a clause  $C$ , a copy or variant of  $C$  is a clause in which all variables of  $C$  have been renamed by fresh variables.

– A clause  $C$  is self-resolving iff the resolution rule can be applied on  $C$  and a copy of  $C$ . One such example is clause 2 of example 5.22.

EXERCISE 5.7.– (soundness of binary resolution). Use the method of semantic tableaux to prove that the binary resolution rule for FOL is correct (i.e. that every model of the parent clauses is a model of the resolvent).  $\square$

REMARK 5.25.– (unsatisfiability and satisfiability by resolution). Each branch in a semantic tree (see section 5.6) denotes a partial interpretation of the set of clauses  $S$ ; if  $S$  is unsatisfiable, then all the branches will falsify an instance of at least one clause in  $S$  (closed tree). By “unfolding” the closed tree (see the correction of exercise 3.34), we can obtain a refutation by resolution of a set of Herbrand instances of  $S$ . The proof is completed by using the so-called “lifting lemma” that enables us to relate closed instantiated clauses (i.e. propositional clauses) and those that are not instantiated. In other words, the following diagram commutes:

$$\begin{array}{ccc} C_1, C_2 & \longrightarrow & \theta C_1, \theta C_2 \\ \downarrow \sigma & & \downarrow \gamma R \\ \sigma C_1, \sigma C_2 & \xrightarrow{R} & \text{Resolvent} \end{array}$$

where  $\sigma$ ,  $\theta$ ,  $\rho$ , and  $\gamma$  denote substitutions ( $\sigma$  and  $\gamma$  are closed substitutions) and  $R$  the operation that yields the resolvent of two clauses.

Soundness and (a consequence of) *refutational completeness* are often presented using operator  $\mathcal{R}$  (see also definition 3.16):

Let  $S$  denote a (finite) set of FOL clauses.

–  $\mathcal{R}(S) = S \cup \{R(C_1, C_2) \mid C_1, C_2 \in S\}$

–  $\mathcal{R}^0(S) = S$

–  $\mathcal{R}^{n+1}(S) = \mathcal{R}(\mathcal{R}^n(S))$  pour  $n \geq 0$

–  $\mathcal{R}^*(S) = \bigcup_{n \geq 0} \mathcal{R}^n(S)$

$\mathcal{R}^*(S)$  denotes the search space, of which all strategies try to explore the smallest subset.

Thus,

– if  $\exists n \geq 0, \square \in \mathcal{R}^n(S)$ , then  $S$  unsatisfiable.

- if  $\exists n \geq 0, \mathcal{R}^{n+1}(S) = \mathcal{R}^n(S)$  and  $\square \notin \mathcal{R}^n(S)$ , then  $S$  **satisfiable**.
- else (FOL *undecidable*)? (This alternative is impossible in PL). □

EXAMPLE 5.22.- Consider the clauses 1, 2, and 3 below. We indicate an application of the resolution rule by specifying, as for PL, the selected clauses and literals, together with the unifier.

- 1)  $P(a)$
  - 2)  $\neg P(x) \vee P(f(x))$
  - 3)  $\neg P(f(f(f(a))))$
  - 4)  $P(f(a))$                        $(1, 1) - (2, 1) \quad \{x \leftarrow a\}$
- 

EXAMPLE 5.23.- (what should not be done). We may wonder why it is necessary to apply the method strictly if “we can see” the replacements that can be done to apply the resolution rule for PL, in other words, whether we can get rid of unification:

- 1)  $P(a)$
- 2)  $\neg P(x) \vee P(f(x))$
- 3)  $\neg P(f(f(f(a))))$
- 4)  $\neg P(a) \vee P(f(a))$  in 2:  $\{x \leftarrow a\}$
- 5)  $P(f(a))$                        $(1, 1) - (4, 1)$

This way of proceeding is correct from a logical point of view (variables in clauses are universally quantified and can be replaced by any constant), but it is incorrect if you are asked to apply the resolution rule in FOL.

Furthermore, it neglects the most original and powerful characteristic of the resolution rule: the presence of the unification algorithm.

Perhaps, you can convince yourselves by trying to “see” what replacements should be performed to apply resolution between clauses 2 and 3 below:

- 1)  $P(a)$
  - 2)  $\neg P(x) \vee Q(g(x, x))$
  - 3)  $\neg Q(g(h(y), y))$
- 

EXAMPLE 5.24.- (grandparents).

Every human has a parent:

i)  $\forall x \exists y P(y, x)$

The parent of a parent is a grandparent:

$$\text{ii) } \forall v \forall u \forall w [P(u, v) \wedge P(v, w) \Rightarrow G(u, w)]$$

Every human has a grandparent:

$$\text{iii) } \forall z \exists s G(s, z)$$

We want to use the resolution rule to show that (iii) is a logical consequence of (i) and (ii). We therefore negate and Skolemize (iii):  $\neg \forall z \exists s G(s, z) \rightsquigarrow \exists z \forall s \neg G(s, z) \rightsquigarrow \forall s \neg G(s, a)$ . It is also necessary to Skolemize (i).

We consider the set of clauses 1, 2, 3 below.

$$1) P(f(x), x)$$

$$2) \neg P(u, v) \vee \neg P(v, w) \vee G(u, w)$$

$$3) \neg G(s, a)$$

and we provide a refutation for this set:

$$4) \neg P(v, w) \vee G(f(v), w) \quad (1, 1) - (2, 1) \quad \{u \leftarrow f(v)\}$$

% formally speaking, the unifier is  $\{u \leftarrow f(v), x \leftarrow v\}$ , but as the couple  $x \leftarrow v$  is not useful in what follows, it is not included.

$$5) G(f(f(w)), w) \quad (1, 1) - (4, 1) \quad \{v \leftarrow f(w)\}$$

$$6) \square \quad (5, 1) - (3, 1) \quad \{s \leftarrow f(f(a)), w \leftarrow a\}. \quad \square$$

EXAMPLE 5.25.- (natural numbers). The following set of clauses translates the statement 0 is a natural number and if  $x$  is a natural number, then so is the successor of  $x$ :

$$\begin{aligned} & N(a) \\ & \neg N(x) \vee N(s(x)) \end{aligned}$$

We want to use the resolution rule to show that 3 is a natural number. We therefore add the following clause to the two previous ones:

$$\neg N(s(s(s(a))))$$

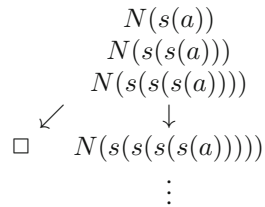
and we try to derive  $\square$ .

a) Backward chaining:

$$\begin{aligned} & \neg N(s(s(s(a)))) \\ & \neg N(s(s(a))) \\ & \neg N(s(a)) \\ & \neg N(a) \\ & \square \end{aligned}$$



b) Forward chaining:



□

EXAMPLE 5.26.— (first negate and then Skolemize). Consider the following reasoning, formalized in FOL:

- 1)  $\forall x \exists y Q(x, y)$
  - 2)  $\forall x \forall y. Q(x, y) \Rightarrow P(x, y)$
- 
- 3)  $\forall x \exists y P(x, y)$

Can you prove that this reasoning is correct (or incorrect)?

To prove that the reasoning is correct (or incorrect)

**using the resolution method**

Mr (or Mrs) A proposes:

Skolemize the conclusion, then negate the result, which leads, say, to formula (3'), and then try to prove the unsatisfiability of the set of clauses obtained from (1), (2), and (3')

and

Mr (or Mrs) B proposes:

Negate the conclusion, then Skolemize the result, which leads, say, to formula (3''), and then try to prove the unsatisfiability of the set of clauses obtained from (1), (2), and (3'').

Who is right?

- a) Mr (or Mrs) A
- b) Mr (or Mrs) B
- c) Both of them
- d) None of them

i) We use the method proposed by Mr (or Mrs) A

$$\forall z \exists y P(z, y) \xrightarrow{skol.} \forall z P(z, f(z)) \xrightarrow{neg.} \exists z \neg P(z, f(z)) \xrightarrow{skol.} \neg P(a, f(a))$$

The set of clauses to refute is thus 1. to 3.:

- 1)  $Q(x, g(x))$
- 2)  $\neg Q(u, v) \vee P(u, v)$
- 3)  $\neg P(a, f(a))$
- 4)  $P(x, g(x)) \quad (1, 1) - (2, 1) \{u \leftarrow x; v \leftarrow g(x)\}$
- 5)  $\neg Q(a, f(a)) \quad (3, 1) - (2, 2) \{u_1 \leftarrow a; v_1 \leftarrow g(a)\}$

The resolution rule can no longer be applied; hence  $\{1., 2., 3.\}$  is *sat* and as a consequence, the reasoning is *incorrect* (according to Mr. (or Mrs) A)

ii) We use the method proposed by Mr (or Mrs) B

$$\forall z \exists y P(z, y) \xrightarrow{neg.} \exists z \forall y \neg P(z, y) \xrightarrow{skol.} \forall y \neg P(a, y)$$

The set of clauses to refute is thus 1. to 3.:

- 1)  $Q(x, g(x))$
- 2)  $\neg Q(u, v) \vee P(u, v)$
- 3)  $\neg P(a, y)$
- 4)  $P(x, g(x)) \quad (1, 1) - (2, 1) \{u \leftarrow x; v \leftarrow g(x)\}$
- 5)  $\square \quad (3, 1) - (4, 1) \{x \leftarrow a; y \leftarrow g(a)\}$

We have proved that the set of clauses  $\{1., 2., 3.\}$  is *unsat*, and as a consequence, the reasoning is *correct* (according to Mr (or Mrs) B)

**Mr (or Mrs) B is right.**

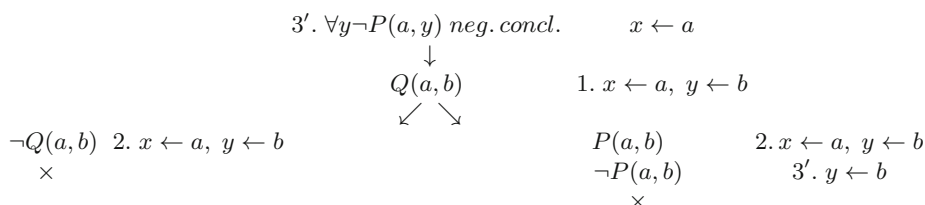
Why is that?

Proof in natural language

$Q$  denotes a total relation (1.), i.e. for any object  $x$ , there exists an object  $y$  that is related to  $x$  in  $Q$ . Two objects that are in the relation denoted by  $Q$  are also in the relation denoted by  $P$  (2.). Hence, for any object  $x$ , there exists an object  $y$  that is related to  $x$  in  $P$ .

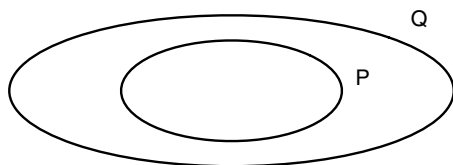
Formal proof (semantic tableaux)

We consider the set of formulas  $\{\forall x\exists yQ(x, y), \forall x\forall y.Q(x, y) \Rightarrow P(x, y), \exists x\forall y\neg P(x, y)\}$

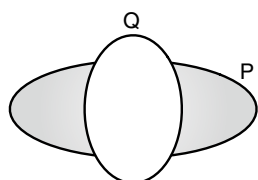


**Explanation**

If formula  $Q$  is a logical consequence of formula  $P$ , then the relationship between their models is:



But as Skolemization can lead to the “loss of models”, after it is performed, we may get the following relationship between the models of the premises ( $P$ ) and the conclusion ( $Q$ ).



The shadowed zone in  $P$  corresponds to the counter examples. □

EXERCISE 5.8.– Use the resolution method to prove that the following reasoning is correct.

- 1)  $\exists x[P(x) \wedge \forall y(R(y) \Rightarrow S(x, y))]$
  - 2)  $\forall x[P(x) \Rightarrow \forall y(Q(y) \Rightarrow \neg S(x, y))]$
- 
- 3)  $\forall x[R(x) \Rightarrow \neg Q(x)]$  □

EXAMPLE 5.27.– Consider the set of clauses 1 to 7 below.

We show that this set is unsatisfiable by using a unit strategy (at least one of the parents is a unit clause). The underlying idea is that the obtained resolvent contains

strictly less literals than the non-unit parent clause, and the rule thus generates clauses that are potential candidates for the generation of  $\square$ .

- |  |  |
|--|--|
| 1) $S(x_1, f(x_1)) \vee V(x_1) \vee \neg E(x_1)$ |  |
| 2) $C(f(x_2)) \vee V(x_2) \vee \neg E(x_2)$      |  |
| 3) $P(a)$  |  |
| 4) $E(a)$  |  |
| 5) $P(x_5) \vee \neg S(a, x_5)$                  |  |
| 6) $\neg P(x_6) \vee \neg V(x_6)$                |  |
| 7) $\neg P(x_7) \vee \neg C(x_7)$                |  |
| 8) $\neg V(a)$                                   | (3, 1) – (6, 1) $\{x_6 \leftarrow a\}$                 |
| 9) $C(f(a)) \vee V(a)$                           | (2, 3) – (4, 1) $\{x_2 \leftarrow a\}$                 |
| 10) $C(f(a))$                                    | (8, 1) – (9, 2)  |
| 11) $S(a, f(a)) \vee V(a)$                       | (1, 3) – (4, 1) $\{x_1 \leftarrow a\}$                 |
| 12) $S(a, f(a))$                                 | (8, 1) – (11, 2)                                       |
| 13) $P(f(a))$                                    | (5, 2) – (12, 1) $\{x_5 \leftarrow f(a)\}$             |
| 14) $\neg C(f(a))$                               | (7, 1) – (13, 1) $\{x_7 \leftarrow f(a)\}$             |
| 15) $\square$                                    | (10, 1) – (14, 1) <span style="float: right;">□</span> |

EXAMPLE 5.28.– (a theorem in group theory). We use the resolution rule to show that *every element of a group has a right inverse*.

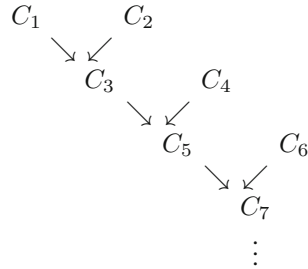
Clauses 1 to 6 below are the clausal form of the set of axioms that define a group and of the negation of the conclusion.

We apply a backward chaining strategy, meaning that we begin by applying the resolution rule with the negation of the conclusion as one of the parent clauses, and we proceed with a *linear strategy*, meaning that one of the parent clauses is always the last clause obtained.

This is very close to what a human would do: conclusions (lemmas) do not need to be memorized for future usage.

Similar to what a program would do, we have systematically renamed variables to avoid any confusion.

A linear strategy is often represented graphically as follows:



- 1)  $P(x_1, y_1, f(x_1, y_1))$
- 2)  $\neg P(x_2, y_2, u_2) \vee \neg P(y_2, z_2, v_2) \vee \neg P(u_2, z_2, w_2) \vee P(x_2, v_2, w_2)$
- 3)  $\neg P(x_3, y_3, u_3) \vee \neg P(y_3, z_3, v_3) \vee \neg P(x_3, v_3, w_3) \vee P(u_3, z_3, w_3)$
- 4)  $P(a, y_4, y_4)$
- 5)  $P(g(u_5), u_5, a)$
- 6)  $\neg P(x_6, h(x_6), h(x_6)) \vee \neg P(k(x_6), z_6, x_6)$
- 7)  $\neg P(k(a), z_7, a)$  (4, 1) – (6, 1)  $\{x_6 \leftarrow a, y_4 \leftarrow h(a)\}$
- 8)  $\neg P(x_8, y_8, k(a)) \vee \neg P(y_8, z_8, v_8) \vee \neg P(x_8, v_8, a)$  (3, 4) – (7, 1)  
 $\{u_3 \leftarrow k(a), w_3 \leftarrow a\}$
- 9)  $\neg P(g(v_9), y_9, k(a)) \vee \neg P(y_9, z_9, v_9)$  (8, 3) – (5, 1)  $\{x_8 \leftarrow g(u_8)\}$
- 10)  $\neg P(g(v_{10}), a, k(a))$  (9, 2) – (4, 1)  $\{y_9 \leftarrow a\}$
- 11)  $\neg P(g(v_{11}), y_{11}, u_{11}) \vee \neg P(y_{11}, z_{11}, a) \vee \neg P(u_{11}, z_{11}, k(a))$   
(10, 1) – (2, 4)  $\{x_2 \leftarrow g(v_{10}), v_2 \leftarrow a, w_2 \leftarrow k(a)\}$
- 12)  $\neg P(g(v_{12}), y_{12}, a) \vee \neg P(y_{12}, k(a), a)$  (11, 3) – (4, 1)  $\{u_{11} \leftarrow a, z_g(k(a))\}$
- 14)  $\square$  (13, 1) – (5, 1)  $\{u_5 \leftarrow g(k(a)), v_{13} \leftarrow g(k(a))\}$  □

EXAMPLE 5.29.– (the monkey and the banana). A monkey wants to eat a banana that is hanging from the ceiling of a room. The monkey is too small to reach the banana. However, the monkey can walk in the room, carry a chair that is in the room, climb on the chair and take the banana to eat it.

We want to describe this situation with a set of clauses. Functional terms can be used to denote actions.

The question will be: does there exist a state that is reachable from a given initial state, where the monkey can catch the banana?

In order to solve this problem using resolution, we choose the following particularly simple formalization (set of Horn clauses, see section 3.9).

$P(x, y, z, s)$ : in state  $s$ , the monkey is at position  $x$ , the banana at position  $y$ , and the chair at position  $z$ .

$R(s)$ : in state  $s$ , the monkey can reach the banana.

$f(x, y, s)$ : the state that is reached from state  $s$ , if the monkey walks from  $x$  to  $y$ .

$h(s)$ : the state reached if the monkey is in state  $s$  and climbs on the chair.

$P(a, b, c, d)$ : the initial state.

The corresponding set of clauses is:

$$1) \neg P(x, y, z, s) \vee P(z, y, z, f(x, z, s))$$

$$2) \neg P(x, y, x, s) \vee P(y, y, y, g(x, y, s))$$

$$3) \neg P(b, b, b, s) \vee R(h(s))$$

$$4) P(a, b, c, d)$$

The question is  $\exists u R(u)$ , which will be negated to try to derive  $\square$ , thus yielding the clause  $\neg R(u)$ .

Sometimes, the question is  $\neg R(u) \vee Answer(u)$  to explicitly retrieve the answer (see solution to exercise 5.9).  $\square$

EXERCISE 5.9.– Find the solution to the problem of the monkey and the banana by applying the resolution rule and a linear strategy.  $\square$

EXERCISE 5.10.–

a) Prove by resolution that the set of clauses below is unsatisfiable:

$$\neg P(x) \vee Q(x) \vee R(x, f(x))$$

$$\neg P(u) \vee Q(u) \vee S(f(u))$$

$$T(a)$$

$$P(a)$$

$$\neg R(a, y) \vee T(y)$$

$$\neg T(z) \vee \neg Q(z)$$

$$\neg T(w) \vee \neg S(w)$$

b) Consider the set of clauses below:

$$1) P(x) \vee P(a) \vee R(x)$$

$$2) \neg P(y) \vee Q(y)$$

$$3) \neg R(z) \vee Q(z) \vee M(z)$$

$$4) \neg Q(w) \vee \neg R(w)$$

$$5) \neg M(v) \vee \neg R(v) \vee \neg R(a)$$

Is it possible to prove by resolution:

- i) that it is unsatisfiable?
- ii) that it is satisfiable?
- iii) we cannot say anything? □

**5.7.1. Variables must be renamed**

The clause  $\boxed{\forall x} (P(x) \vee \neg Q(y))$  is equivalent on the Herbrand universe (see definition 5.9) on  $\Sigma = \{a, b, f^{(1)}\}$  to

$$\{P(a) \vee \neg Q(b), P(f(a)) \vee \neg Q(f(f(b))), \dots\}.$$

But if we want to discover the instances of interest by unification, it is better to write:

$$\{\sigma_1(P(x_1) \vee \neg Q(y_1)), \sigma_2(P(x_2) \vee \neg Q(y_2)), \dots\}$$

with:

$$\sigma_1 = \{x_1 \leftarrow a, y_1 \leftarrow b\} \quad \sigma_2 = \{x_2 \leftarrow f(a), y_2 \leftarrow f(f(b))\}, \dots$$

(the  $\sigma_i$  correspond to substitutions that would be computed by the resolution rule)

EXAMPLE 5.30.– We prove by resolution that the set of clauses 1,2,3 below is unsatisfiable:

- 1)  $\neg P(x) \vee P(f(x))$
- 2)  $P(a)$
- 3)  $\neg P(f(f(a)))$
- 4)  $P(f(a))$             (1, 1) – (2, 1)     $\{x \leftarrow a\}$
- 5)  $P(f(f(a)))$         (1, 1) – (4, 1)     $\{x \leftarrow f(a)\}$
- 6)  $\square$                     (3, 1) – (5, 1)

If variables are not renamed in 1, the contradiction cannot be proved (we would have assimilated  $\forall$  with  $\exists$ , i.e. usable only once). □

EXERCISE 5.11.– Is the wff  $\exists x \forall y P(x, y)$  a logical consequence of  $\forall x \exists y P(x, y)$ ?  
With symbols:

$$\forall x \exists y P(x, y) \models \exists x \forall y P(x, y)?$$

As usual,  $x$  and  $y$  denote variables and  $P$  a predicate symbol.

Answer using the resolution method. □

EXERCISE 5.12.– ( $\mathcal{S}_1$  and resolution). We have already appreciated the general difficulty of finding a proof in a formal system (for example,  $\mathcal{S}_1$ ). Hence, the idea of trying to do this work automatically by using, for example, the resolution rule.

You are asked:

i) To encode as clauses  $A1$ ,  $A2$ ,  $A3$ , and  $MP$  (see section 3.4) and to prove, using the resolution rule, that:

$$\vdash_{\mathcal{S}_1} A \Rightarrow A$$

ii) To rephrase (as you would, for example, to explain it to someone who does not know the resolution rule) this proof.

Hint: encode  $x \Rightarrow y$  by  $i(x, y)$  and use predicate symbol  $P$  with the meaning  $P(x)$ :  $x$  is provable in  $\mathcal{S}_1$ .  $\square$

### 5.8. A decidable class: the monadic class

FOL is undecidable (but semi-decidable). Finding classes or fragments (i.e. sets of cwwfs that are proper subsets of  $\mathcal{L}_1$  and that are decidable is one of the most important problems of logic, which is known as the “classical decision problem (for FOL)”. It can be presented in different ways.

Given  $F \in \mathcal{L}_1$ , the problem of:

- satisfiability (coherence, consistency): decide if  $F$  is satisfiable;
- validity: decide if  $F$  is valid;
- provability: given a formal system  $\mathcal{S}$  that is correct and complete, decide whether  $\vdash_{\mathcal{S}} F$ .

The interest of being able to characterize such classes (or fragments) is clear: trying to have a “good” expressive power, while retaining “very good” decidability properties.

In this section, we shall prove the decidability of some fragments of FOL. We begin by some definitions.

DEFINITION 5.15.– (finitely controllable class, finite model property). *A class  $\mathcal{C}$  is finitely controllable or has the finite model property iff for all  $F \in \mathcal{C}$ , if  $F$  is satisfiable, then  $F$  admits a finite model.*

REMARK 5.26.– Every finitely controllable class is decidable. The converse is not true: there are decidable classes that do not have the finite model property.  $\square$

Among the finitely controllable class, we shall study *monadic logic*.



DEFINITION 5.16.– (monadic class of FOL). *The set of cwffs of  $\mathcal{L}_1$  containing exclusively unary predicates (and possibly =), but not containing any functional symbols (in particular, constants) is called the monadic class of FOL (or monadic FOL), denoted by  $MFOL^=$ .*

*If the cwffs do not contain the equality symbol =, then we have the pure monadic class, denoted by  $MFOL$ .*

*$MFOL$  was used to formalize Aristotle's syllogistic. See also digression 8.1.*

The proof of the decidability of  $MFOL^=$  is based on the following remarks:

1) To test the satisfiability of a wff of FOL (see definition 5.6), what matters is the *cardinality* of the domain of discourse  $D$  of the potential model (the definition of satisfiability *does not take the names of the elements of  $D$  into account*);

2) On a finite domain, say  $\{a_1, \dots, a_n\}$ ,  $\forall xP(x)$  is equivalent to  $\bigwedge_{i=1}^n P(a_i)$  and  $\exists xP(x)$  is equivalent to  $\bigvee_{i=1}^n P(a_i)$ .

3) Key point: (theorem 5.11): proving that  $MFOL^=$  is finitely controllable by providing an upper bound on the cardinality of the domains on which it suffices to test the satisfiability of the formulas in the class.

4) On a finite domain, we can consider all possible evaluations of unary predicates (and equality between variables) and use property 2 above. There are therefore only a finite number of interpretations to test.

The following theorem proves to be the key property of the decision procedure for  $MFOL^=$ .

THEOREM 5.11.– ( $MFOL^=$  has the finite model property). *A cwff  $F$  of  $MFOL^=$  containing  $k$  predicate symbols and  $v$  variables is satisfiable (on a structure  $\mathcal{M} = \langle D; \mathcal{R} \rangle$ ) iff  $F$  is satisfiable on a structure*

$\mathcal{M}' = \langle D_{max}; \mathcal{R}' \rangle$  with:

$$D_{max} \leq 2^k \times v \quad (k \geq 0, v \geq 1)$$

PROOF.– (detailed outline).

*if:*

Trivial: if a wff is satisfiable on a domain of a given cardinality, then it is satisfiable.

*only if:*

Define relation  $R$  on  $D$ :

for  $a_1, a_2 \in D$

$a_1 R a_2$  iff:

$$P_1^{\mathcal{M}}(a_1) = P_1^{\mathcal{M}}(a_2) \text{ and } P_2^{\mathcal{M}}(a_1) = P_2^{\mathcal{M}}(a_2) \text{ and } \dots \text{ and } P_k^{\mathcal{M}}(a_1) = P_k^{\mathcal{M}}(a_2)$$

where the  $P_i$ 's ( $1 \leq i \leq k$ ) denote the  $k$  predicates in  $F$  and, as usual,  $P_i^{\mathcal{M}}(a_j)$  ( $j = 1, 2$ ) **T** or **F** depending on whether  $P_i$  is evaluated to **T** or **F** on  $a_j$ .

If we were to rigorously stick to definition 5.6, we would have to say that  $a_1, a_2$  are in relation  $R$  if and only if they belong (or do not belong) to the relation assigned by the interpretation (model) to  $P_i$  ( $1 \leq i \leq k$ ), i.e.

$$P_i^{\mathcal{M}}(a_1) \text{ iff } P_i^{\mathcal{M}}(a_2) \quad (1 \leq i \leq k)$$

% The idea is that the elements of  $D$  on which the predicates are evaluated to the same truth value are regrouped.

$R$  is an equivalence relation (see definition 3.26).

The proof is trivial and is a result of the properties of equality (see also section 9.1.4):

- reflexive  $a_1 R a_1$ , as  $P_i^{\mathcal{M}}(a_1) = P_i^{\mathcal{M}}(a_1)$  ( $1 \leq i \leq k$ );
- symmetric: if  $a_1 R a_2$  then  $a_2 R a_1$ , as if  $P_i^{\mathcal{M}}(a_1) = P_i^{\mathcal{M}}(a_2)$ , then  $P_i^{\mathcal{M}}(a_2) = P_i^{\mathcal{M}}(a_1)$  ( $1 \leq i \leq k$ );
- transitive: if  $a_1 R a_2$  and  $a_2 R a_3$ , then  $a_1 R a_3$ , as if  $P_i^{\mathcal{M}}(a_1) = P_i^{\mathcal{M}}(a_2)$  and  $P_i^{\mathcal{M}}(a_2) = P_i^{\mathcal{M}}(a_3)$ , then  $P_i^{\mathcal{M}}(a_1) = P_i^{\mathcal{M}}(a_3)$  ( $1 \leq i \leq k$ ).

As there are two possibilities for each  $P_i$  (for any  $a_j$ ), there are  $2^k$  equivalence classes.

If we now consider the equalities, that can only relate the  $v$  variables (see definition 5.16), it is sufficient, for each equivalence class, to consider at most  $a_1, a_2, \dots, a_v$  elements ( $a_i \in D$ ;  $1 \leq i \leq v$ ). If the equivalence class contains  $p$  ( $p < v$ ) elements, we consider  $a_1, a_2, \dots, a_p$ .

We thus consider:

$\mathcal{M}' = \langle D_{max}; \mathcal{R}' \rangle$ ; with:

$$D_{max} = \bigcup_{i=1}^{2^k} \bigcup_{j=1}^{p \leq v} a_j^{(i)}$$

i.e.  $\text{card}(D_{max}) \leq 2^k \times v$ .

We define the relations in  $\mathcal{R}'$ :

$P_i^{\mathcal{M}}([a])$  iff  $P_i^{\mathcal{M}}(a)$ , where  $[a]$  is (a representative of) the equivalence class of  $a$  ( $a \in D$ ).

% Recall that equivalence classes of  $D$  form a partition of  $D$ .

By structural induction (see section 5.2.3) and using properties 1 to 4 of section 5.8, we prove:

$\models_{\mathcal{M}} F$  iff  $\models_{\mathcal{M}'} F$ . □

REMARK 5.27.– (syllogistic and decidability). Aristotelian syllogisms can be formalized in MFOL (see remark 2.7). As this is a decidable fragment of FOL and as the latter is undecidable, syllogistic is not sufficient to reason in FOL. □

EXAMPLE 5.31.– (see also example 5.11).  $F: \exists xP(x) \Rightarrow \forall xP(x)$

Here,  $k = 1$ ;  $v = 1$ ; hence, it suffices to test  $D_{max} = \{a, b\}$ .

Indeed, we consider  $\neg F$ , and on  $D_{max}$ , we can obtain the models of  $\neg F$  (i.e. counter examples of  $F$ ):  $\{\neg P(a), P(b)\}$  and  $\{P(a), \neg P(b)\}$ . Hence,  $F$  is not valid. □

EXAMPLE 5.32.– (some cwffs in the monadic class). We have treated some of those by the method of semantic tableaux in example 5.13, exercise 5.3 (a)-(b)-(h)-(k2), and exercise 5.4.

Note that, in general, we can detect (non)-validity on universes of a lesser cardinality than the upper bound of theorem 5.11. □

### 5.8.1. Some decidable classes

There exist techniques other than the one we used to solve the classical decision problem for MFOL $^=$ . These techniques are beyond the scope of this work.

The decidable fragments of FOL are characterized by the prefixes of the prenex normal form that is equivalent to the cwff that is provided as an input (see theorem 5.3).

– *Bernays–Schönfinkel*:  $\exists x_1 \dots \exists x_m \forall y_1 \dots \forall y_n. M$  (also denoted by  $\exists^* \forall^*$ ;  $M$  in MFOL);

– *Ackermann*:  $\exists x_1 \dots \exists x_m \forall y \exists z_1 \dots \exists z_n. M$  (also denoted by  $\exists^* \forall \exists^*$ );

– *Herbrand*: arbitrary prefix and matrix  $M : \bigwedge_{i=1}^n L_i$  with  $L_i$  ( $1 \leq i \leq n$ ): literals;

– *Gödel–Kalmar–Schütte*:  $\exists x_1 \dots \exists x_m \forall y_1 \forall y_2 \exists z_1 \dots \exists z_n. M$  (also denoted by  $\exists^* \forall^2 \exists^*$ );

–  *$\forall x \exists y \forall z$ -Horn*:  $\forall x \exists y \forall z. M_H$  with  $M_H : \bigwedge_{i=1}^n L_i$  and  $L_i$  ( $1 \leq i \leq n$ ): disjunction of literals with at most one positive literal.

### 5.9. Limits: Gödel's (first) incompleteness theorem

A particularly simple proof of this capital theorem (given by G. Boolos) uses the following notions: paradox, arithmetic, truth, proof, algorithm and encoding.

#### Paradox

Here it will be the Berry's paradox, see example 2.1.

#### Arithmetic

See example 3.8.

#### Truth

See section 5.2.1.

#### Proof

See section 3.3.

#### Algorithm

Informally, an algorithm is any well-defined computation procedure that admits a (set of) value(s) as an *input* and produces a (set of) value(s) as an *output*. Examples are a program (in a programming language), a Turing machine, Markov algorithms, formal systems for arithmetic, etc.

#### Encoding (in arithmetic)

*Encoding* a text consists in replacing a word or sentence by another word, a number or a symbol. In cryptography, a *code* uses the substitution at the word level, whereas a *cipher* uses the substitution at the level of letters. The goal is to hide information.

We shall use a translation that associates to every wff of a given language an integer (i.e. a word in arithmetic). The goal is not to hide information but to change its *representation*.

Arithmetic enables us to uniquely encode into a natural number any sequence of symbols (i.e. a string on a vocabulary), for example, a wff in  $\mathcal{L}_1$ , instructions of a Turing machine (or of a program), a computation of a Turing machine (as it can be described as a sequence of instantaneous descriptions of the Turing machine), proofs in a formal system, etc.

The first to propose an encoding was K. Gödel (in 1931), the natural number encoding an expression  $M$  has since been known as the *Gödel number of  $M$* , denoted by  $gn(M)$ .

DEFINITION 5.17.– *Let  $M$  denote an expression (a word) on a vocabulary  $V$ .*

*We assign to each symbol in  $V$  an odd number ( $> 1$ ):*

$$f : V \longrightarrow \mathbb{N} \setminus \{0, 1, 2\}$$

*Given the expression  $M : \gamma_1\gamma_2\dots\gamma_n$*

$$f(\gamma_i) = a_i ; a_i \text{ even} \quad (1 \leq i \leq n)$$

*the Gödel number of  $M$  is the natural number:*

$$ng(M) = \prod_{k=1}^n \text{Prime}(k)^{a_k}$$

*where  $\text{Prime}(k)$  is the  $k^{\text{th}}$  prime number.*

*If  $M = \epsilon$  (i.e. the empty string), then  $ng(\epsilon) = 1$ .*

For example, for a string  $M : \gamma_1\gamma_2\gamma_3\gamma_4\gamma_5$ ,  $ng(M) = 2^{a_1} \times 3^{a_2} \times 5^{a_3} \times 7^{a_4} \times 11^{a_5}$

Why this encoding? The justification is immediate if we recall the fundamental theorem of arithmetic (see below).

THEOREM 5.12.– (fundamental theorem of arithmetic). *Every  $x \in \mathbb{N}$  ( $x > 1$ ) can be represented as*

$$p_1^{m_1} \times p_2^{m_2} \times \dots \times p_k^{m_k}$$

*with  $p_i$  ( $1 \leq i \leq k$ ): prime and  $p_i \neq p_j$  (for all  $i \neq j$ ).*

*This representation is unique (up to the commutativity of  $\times$ ).*

An immediate corollary is:

COROLLARY 5.2.– (uniqueness of the encoding of words). *If  $ng(M) = ng(N)$ , then  $M = N$ .*  $\square$

DEFINITION 5.18.– (Gödel number of sequences of words). *Let  $\Gamma : M_1M_2\dots M_n$  denote a finite sequence of expressions (i.e. words on a vocabulary), the Gödel number of  $\Gamma$  is defined as:*

$$gn(\Gamma) = \prod_{k=1}^n \text{Prime}(k)^{ng(M_k)}.$$

An immediate corollary is the following:

COROLLARY 5.3.– (uniqueness of the encoding of sequences of words). *If  $ng(\Gamma_1) = ng(\Gamma_2)$ , then  $\Gamma_1 = \Gamma_2$ .*  $\square$

REMARK 5.28.– (other encodings). A word or a sequence of words has a unique  $gn$ , but a set of  $n$  words (expressions) has  $n!$  possible  $gn$ . We will always consider *sequences* of words, as they each have a *unique gn*.

Other encodings are possible (and used), for example, the encoding of  $n$ -tuples, which is used in the proof of theorem 5.5. They are also called Gödel numbers.  $\square$

We give a version of Gödel's famous theorem that is particularly interesting in computer science.

THEOREM 5.13.– (Gödel's first incompleteness theorem). *There is no algorithm that can produce all true cwffs of arithmetic (i.e. without there being any false cwff in the list).*

PROOF.– A sensible observation about this theorem is that if such an algorithm were to exist, in order to know whether a conjecture  $Conj$  holds or not in arithmetic, it would suffice to "sit and wait" for  $Conj$  or  $\neg Conj$  to be added to the list.

**Proof technique:** we assume that such an algorithm  $m$  exists and we construct a cwff that is true in arithmetic but is not in the output of  $M$ .

If  $n \in \mathbb{N}$ , we will write  $[n]$  for  $\underbrace{ss\dots s}_n 0$

where  $s$  denotes the successor function and  $0$  denotes itself. % For example,  $4 : ssss0$ .

– We will say that a formula  $F(x)$  designates  $n \in \mathbb{N}$  if the formula  $\forall x(F(x) \leftrightarrow x = [n])$  occurs in the output of  $M$ .

For example, if in the output of  $M$  we find

$$\forall x(x + x = ssss0 \leftrightarrow x = ss0),$$

then formula  $x + x = ssss0$  *designates* the integer 2.

– Names are unique: imagine that this is not the case, for example, if the following cwffs occur in the output of  $M$ :

- 1)  $\forall x(F(x) \leftrightarrow x = [n])$
- 2)  $\forall x(F(x) \leftrightarrow x = [p])$

then, after replacing  $F(x)$  in (2) by the name it designates in (1) we obtain:

$$\forall x(x = [n] \leftrightarrow x = [p]) \text{ hence } [n] = [p], \text{ i.e. } n = p.$$

– If the number of symbols in our formalization of arithmetic is  $A$  (if there are infinitely many variables, they can be generated, e.g. by  $x$  and  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}, x_0, x_1, x_2, x_3, \dots$ ) there are at most  $A^i$  formulas containing  $i$  symbols.

For any  $m \in \mathbb{N}$ , there are therefore finitely many numbers that are designated by formulas with at most  $m$  symbols. Thus, there are integers that are not designated by formulas of at most  $m$  symbols. Hence, there exists a least one integer that is not designated by such a formula (we order finitely many designated numbers and take the successor of the greatest designated number).

– We construct a formula  $C(x, y)$  meaning “ $x$  is a number that is designated by a formula containing  $z$  symbols”:

$$C(x, z) : \underbrace{\forall x(F(x) \leftrightarrow x = [n])}_{\mathcal{F}} \wedge \text{length}(\mathcal{F}) = z$$

$\text{length}$  being, for example, the program from example 6.17 feeding it as an input:  $\forall, \mathbf{x}, (, \mathbf{F}, (, \mathbf{x}, ), \leftrightarrow, \mathbf{x}, =, [, \mathbf{n}, ], )$ .

We could of course have used any other program that computes the length of a list<sup>18</sup>.

– Let  $B(x, y)$  denote the formula with the intended meaning: “ $x$  is designated by a formula containing less than  $y$  symbols”:

$$B(x, y) : \exists z(z < y \wedge C(x, z))$$

$<$  is definable in arithmetic:  $x < y \text{ :def } \exists z(s(z) + x = y)$ .

---

<sup>18</sup> A program can be viewed as a sequence of instructions, each instruction can have a unique code, and the encoding of theorem 5.5 enables us to assign a unique natural number to a program.

– Let  $A(x, y)$  denote the formula with the intended meaning: “ $x$  is the smallest number that is not designated by a formula containing less than  $y$  symbols”:

$$A(x, y) : \neg B(x, y) \wedge \forall u(u < x \Rightarrow B(u, y))$$

– Let  $k$  denote the number of symbols in  $A'(x, y)$ . Of course (by inspection),  $k > 3$  and let  $F(x)$  denote the formula with the intended meaning “ $x$  is the smallest number that cannot be named by a formula containing less than  $10 \times k$  symbols”:

$$F(x) : \exists y(y = [10] \times [k]) \wedge A(x, y)$$

Let us see how many symbols  $F(x)$  contains:

$$[10] : \underbrace{ss\dots s}_0 0 \rightsquigarrow 11 \text{ symbols}$$

$$[k] : \underbrace{ss\dots s}_k 0 \rightsquigarrow k + 1 \text{ symbols}$$

$$A(x, y) : \rightsquigarrow k \text{ symbols}$$

$$F(x) \text{ also contains the symbols } \exists, y, (, y, \times, =, ), \wedge, ) \rightsquigarrow 10 \text{ symbols}$$

$$F(x) \text{ thus contains a total of } k + 22 \text{ symbols and as } k > 3, k + 22 < 10 \times k,$$

thus  $F(x)$  contains less than  $10 \times k$  symbols.

– As mentioned above, there exists a smallest number among those that are not designated by a formula containing less than  $m$  symbols.

Let  $n$  be this number for  $m = 10 \times k$ .

$n$  is not designated by  $F(x)$ , i.e.

$$(\star) \forall x(F(x) \Leftrightarrow x = [n]) \text{ is not among the outputs of } M.$$

But  $(\star)$  is a cwff that is **true**, as  $n$  is the smallest number that is not designated by a formula containing less than  $10 \times k$  symbols.

We have found a **true** cwff (i.e.  $(\star)$ ) that **is not** in the output of  $M$ .

□

REMARK 5.29.– (incompleteness theorem and constructive mathematics). The notion of a proof is essential in constructive mathematics, as it is used to explain the meaning of existence.

From a constructivist point of view, to say that a proposition  $\Phi$  is true is equivalent to saying that we can find a proof of  $\Phi$ . Some authors call this identity the “To assert is to prove” principle:



**A-P:**  $\Phi \Leftrightarrow \exists p (p \text{ is a proof of } \Phi)$

The following problem arises naturally: *how can Tarski and Gödel's results be reconciled with the principles on which constructivism is founded?* In other words: *what are these truths that cannot be proved (as according to **A-P**, truth coincides with proof)?* The answer is that constructivists do not limit themselves to proofs in formal systems, but principle **A-P** refers to proofs that are correct from a constructivist point of view, knowing that provable means provable by any sound means and *not* provable in a given formal system.

There are therefore arithmetic truths that cannot be proved in a formal system, but that can be proved by correct means from a constructivist point of view.  $\square$



## Chapter 6

# Foundations of Logic Programming

### 6.1. Specifications and programming

The most important task of a programmer (at least till now) can be characterized as filling the gap  $g$  below:

Specification of a problem (Cannot be run)  $\longleftarrow g \longrightarrow$  Program (Can be run)

Logic programming (LP) tends to reduce this gap (and possibly obtain  $g = 0$ ) when the specification is given in a logical language (or a language close to a logical one).

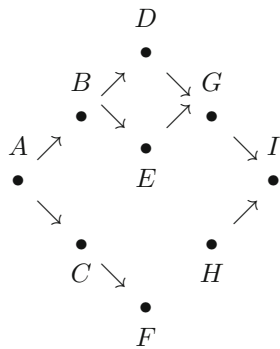
It envisions *calculus* as a *controlled deduction*. This is related to the paradigm:

$$\boxed{\text{algorithm} = \underbrace{\text{logic}}_{\text{what}} + \underbrace{\text{control}}_{\text{how}}}$$

In LP, the emphasis is put on what the program computes and paying as little attention as possible to how it computes. It is a declarative form of programming (as opposed to imperative programming).

We give a first glimpse of such a type of programming.

EXAMPLE 6.1.– Imagine that in the following graph, we want to know whether there are paths from  $A$  to  $I$ , and if so, which ones.



a) We describe the graph using a unary predicate (which corresponds to a property).

$\text{Att}(x)$ :  $x$  is attainable

**Premises**

$\text{Att}(A)$ ; % Must not be forgotten!

$\text{Att}(A) \Rightarrow \text{Att}(B)$ ;

$\text{Att}(B) \Rightarrow \text{Att}(D)$ ;

$\text{Att}(B) \Rightarrow \text{Att}(E)$ ;

$\text{Att}(D) \Rightarrow \text{Att}(G)$ ;

$\text{Att}(E) \Rightarrow \text{Att}(G)$ ;

$\text{Att}(G) \Rightarrow \text{Att}(I)$ ;

$\text{Att}(H) \Rightarrow \text{Att}(I)$ ;

$\text{Att}(A) \Rightarrow \text{Att}(C)$ ;

$\text{Att}(C) \Rightarrow \text{Att}(F)$ ;

The conclusion:

$\text{Att}(I)$ ;

Two essential questions:

- i) which *inference rule(s)* should we use for deduction?
- ii) with what *strategy(ies)* (forward chaining, backward chaining, others)?

In all cases, it is simple to deduce that in this graph, there are two paths from  $A$  to  $I$  paths that cannot be found if  $\text{Att}(A)$  is forgotten.

- b) We describe the graph using a binary predicate (that corresponds to a relation):

$\text{Path}(x, y)$ : there is a path from  $x$  to  $y$

As far as questions (i) and (ii) given above are concerned, we decide<sup>1</sup> to use:

i) the resolution rule (the conclusion is also called *a question*). As usual, the conclusion is negated and we try to derive  $\square$  (contradiction);

ii) backward chaining with the following rules on the stack for the resolvents: we always resolve the first clause in the order 1,2,3... , with the last resolvent obtained (at the beginning, the question) and on the last literal that was pushed (at the beginning, the first literal of the question).

- 1)  $\text{Path}(A, B)$
- 2)  $\text{Path}(A, C)$
- 3)  $\text{Path}(B, D)$
- 4)  $\text{Path}(B, E)$
- 5)  $\text{Path}(C, F)$
- 6)  $\text{Path}(D, G)$
- 7)  $\text{Path}(E, G)$
- 8)  $\text{Path}(G, I)$
- 9)  $\text{Path}(H, I)$
- 10)  $\text{Path}(x, y) \vee \neg\text{Path}(x, z) \vee \neg\text{Path}(z, y)$       % non-elementary path
- 11)  $\neg\text{Path}(A, I)$       % (neg) question
- 12)  $\neg\text{Path}(A, z) \vee \neg\text{Path}(z, I)$       (11, 1) - (10, 1)  $\{x \leftarrow A, y \leftarrow I\}$

---

<sup>1</sup> This decision corresponds to the choice that was made in the most famous of all LP languages: Prolog.

- 13)  $\neg Path(B, I)$  (12, 1) – (1, 1)  $\{z \leftarrow B\}$   
**memorize(12, 1) – (2, 1)**
- 14)  $\neg Path(B, z) \vee \neg Path(z, I)$  (13, 1) – (10, 1)  $\{x_1 \leftarrow B, y_1 \leftarrow I\}$
- 15)  $\neg Path(D, I)$  (14, 1) – (3, 1)  $\{z \leftarrow D\}$   
**memorize(14, 1) – (4, 1)**
- 16)  $\neg Path(D, z) \vee \neg Path(z, I)$  (15, 1) – (10, 1)  $\{x_2 \leftarrow D, y_2 \leftarrow I\}$
- 17)  $\neg Path(G, I)$  (16, 1) – (6, 1)  $\{z \leftarrow G\}$
- 18)  $\square$  (17, 1) – (8, 1)

Note that, as was mentioned in section 5.7.1, each time clause 10 was used, its variables were renamed.

If we want to find all solutions, we need to return to the memorized choices.  $\square$

REMARK 6.1.– (FOL and databases). In a travel agency, the flights that connect cities are represented as a graph, with vertices representing cities and edges representing direct flights between the cities (labelled by, say,  $Flight(\text{city-i}, \text{city-j})$ ).

We will say that a connection between two cities is *acceptable* if we can go from one to the other with at most two stops (i.e. three direct flights).

We want to answer questions of the form:

**Acceptable-flight(city-a, city-b);**

We specify the notion of acceptability in FOL:

$$[\exists z \exists u (Flight(x, z) \wedge Flight(z, u) \wedge Flight(u, y))] \Rightarrow \forall x \forall y \text{Acceptable-flight}(x, y)$$

by applying the rules used in the proof of theorem 5.3 to transform a formula into a clausal form, we obtain:

$$\neg[\exists z \exists u (Flight(x, z) \wedge Flight(z, u) \wedge Flight(u, y))] \vee \forall x \forall y \text{Acceptable-flight}(x, y) \rightsquigarrow$$

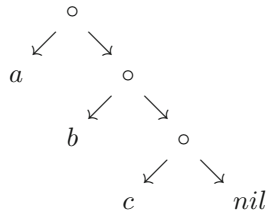
$$\forall z \forall u [\neg Flight(x, z) \vee \neg Flight(z, u) \vee \neg Flight(u, y)] \vee \forall x \forall y \text{Acceptable-flight}(x, y) \rightsquigarrow$$

$$\forall z \forall u \forall x \forall y [\neg Flight(x, z) \vee \neg Flight(z, u) \vee \neg Flight(u, y) \vee \text{Acceptable-flight}(x, y)]$$

which is a Horn clause. By adding this clause to the program describing the graph of flights, we could answer any question on acceptable flights.  $\square$

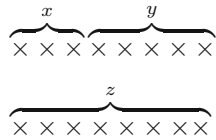
EXAMPLE 6.2.— We proceed with a well-known example for computer scientists: *appending* two lists.

We will use the fact that every list is a tree, for example, the list  $[a, b, c]$  is represented by the tree:

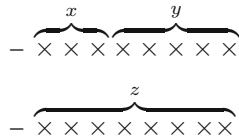


where  $\circ$  denotes the list constructor *cons*; for example,  $\text{cons}(a[b, c, d]) = [a, b, c, d]$  and of course, *nil* denotes the empty list.

Predicates (and logical formulas in general) do not deliver values (as functions do). When we want to mention an object (such as the list obtained by appending the list called  $x$  and the list called  $y$ ), we need to name it:



The program also needs to know that if we add an element to the head of list  $x$ , then it will be at the head of list  $z$ :



The logic program for *append* is the following (with  $\circ$  replaced by  $f$  for the sake of notational coherence, see definition 4.1).

- 1)  $\forall x.\text{append}(\text{nil}, x, x)$
- 2)  $\forall x\forall y\forall z\forall u.\neg\text{append}(x, y, z) \vee \text{append}(f(u, x), y, f(u, z))$

2 reads: if appending list  $x$  to list  $y$  yields list  $z$ , then appending  $\text{cons}(u, x)$  to  $y$  yields the list  $\text{cons}(u, z)$ .

Similar to resolution, *we do not write* the prefixes:

$\forall x$  and

$$\boxed{\forall x \forall y \forall z \forall u}$$

that are always implicit.

We now ask the question

$$\text{append}(\overbrace{f(a, f(b, nil))}^{[a,b]}, \overbrace{f(c, nil)}^{[c]}, \overbrace{f(a, f(b, f(c, nil)))}^{[a,b,c]}) ?$$

We prove that the answer is yes for the given specification of *append* by *deducing* it by resolution (after having negated the conclusion, i.e. the question):

$$\begin{array}{l} 3) \quad \neg \text{append}(f(a, f(b, nil)), f(c, nil), f(a, f(b, f(c, nil)))) \\ 4) \quad \neg \text{append}(f(b, nil), f(c, nil), f(b, f(c, nil))) \\ (3, 1) - (2, 2) \quad \{u \leftarrow a, x \leftarrow f(b, nil), y \leftarrow f(c, nil), z \leftarrow f(b, f(c, nil))\} \\ 5) \quad \neg \text{append}(nil, f(c, nil), f(c, nil)) \\ (4, 1) - (2, 2) \quad \{u_1 \leftarrow b, x_1 \leftarrow nil, y_1 \leftarrow f(c, nil), z_1 \leftarrow f(c, nil)\} \\ 6) \quad \square \\ (5, 1) - (1, 1) \quad \{x_2 \leftarrow f(c, nil)\} \end{array}$$

□

Logic as a programming language is not limited to answering *yes* or *no*, it also has the usual capabilities of languages.

The question we will ask will be:

$$\boxed{\text{Does there exist a list that is in relation via } \textit{append} \text{ with lists } [a, b] \text{ and } [c] ?}$$

EXAMPLE 6.3.– Instead of the resulting list, we write a variable in which we will recover (if it exists) the list mentioned in the question above.

$$\begin{array}{l} 3) \quad \neg \text{append}(f(a, f(b, nil)), f(c, nil), w) \\ 4) \quad \neg \text{append}(f(b, nil), f(c, nil), z) \\ (3, 1) - (2, 2) \quad \{u \leftarrow a, x \leftarrow f(b, nil), y \leftarrow f(c, nil), w \leftarrow f(a, z)\} \\ 5) \quad \neg \text{append}(nil, f(c, nil), z_1) \\ (4, 1) - (2, 2) \quad \{u_1 \leftarrow b, x_1 \leftarrow nil, y_1 \leftarrow f(c, nil), z \leftarrow f(b, z_1)\} \\ 6) \quad \square \\ (5, 1) - (1, 1) \quad \{x_2 \leftarrow f(c, nil), z_1 \leftarrow f(c, nil)\} \end{array}$$

□

REMARK 6.2.– As a side effect of deduction, that answers *yes*, *such an object exists*, the unification algorithm used in the resolution rule allows us to construct the list we were looking for:

$$w = f(a, z) = f(a, f(b, z_1)) = f(a, f(b, f(c, nil))) = [a, b, c]$$

□



REMARK 6.3.– Since we are specifying *relations*, the distinction between input and output parameters (as for functions) is of no interest. It is thus natural to ask questions such as:

Do there exist lists  $x$  and  $y$  that are in relation via *append* with the list  $[a, b, c]$ ?

We obtain as an answer:

$$\begin{array}{ll}
 x = nil & y = [a, b, c] \\
 x = [a] & y = [b, c] \\
 x = [a, b] & y = [c] \\
 x = [a, b, c] & y = nil
 \end{array}
 \quad \square$$

### 6.2. Toward a logic programming language

We also introduce a syntax that is commonly used in practice (others exist):

Logic syntax	Logic programming syntax
$A$	$A \rightarrow;$ or $A;$
$A \vee \neg B_1 \vee \dots \vee \neg B_n$	$A \rightarrow B_1 \dots B_n;$
$\neg B_1 \vee \dots \vee \neg B_n$	$\rightarrow B_1 \dots B_n;$

The intuitive meaning being:

$A \rightarrow;$  means  $A$  is given or, to prove  $A$  there is nothing to do.

$A \rightarrow B_1 \dots B_n;$  means to prove (solve the problem)  $A$  it suffices to prove (solve) the *goals*  $B_1 \dots B_n$ .

$\rightarrow B_1 \dots B_n;$  means we must prove the *goals*  $B_1 \dots B_n$ .

Proving a conclusion (answering a question) boils down to finding  $\square$ , or, equivalently, *deleting all the goals of the question*.

In a clause (also called a rule):

$$C: A \rightarrow B_1 \dots B_n$$

$A$  is the head of  $C$  or left-hand side of  $C$ ,

$B_1 \dots B_n$  the tail or body of  $C$  or right-hand side of  $C$ .

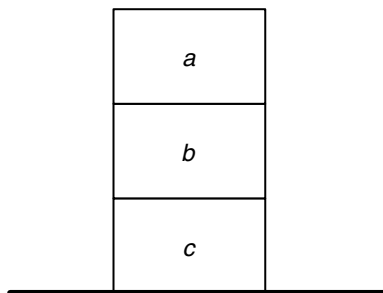
Procedure LP can be considered as an abstract interpreter of the Prolog language.

From now on, we will assume that all logic programs are run with LP.

EXERCISE 6.1.— The strategy that is used by LP may provoke surprises. These “traps” are easy to correct when we are aware of them. This is the object of the following exercise.

a) In what is called the cube world, we define the relations on  $\text{top}(x, y)$  and  $\text{above}(x, y)$ .

We assume the following state of the world:



The following *specification* is proposed for this world:

$\text{on top}(a, b) \rightarrow$ ; % i.e.  $a$  is on top of  $b$

$\text{on top}(b, c) \rightarrow$ ; % i.e.  $b$  is on top of  $c$

$\text{above}(x, y) \rightarrow \text{on top}(x, y)$ ; % if  $x$  is on top of  $y$ , then  $x$  is above  $y$ .

But this specification is *incomplete*, as a cube can be *above* another without being *on top* of it (for example,  $a$  is above  $c$ ).

Which one of (\*) or (\*\*) below must complete the program?

(\*)  $\text{above}(u, v) \rightarrow \text{above}(u, z) \text{ on top}(z, v)$ ;

(\*\*)  $\text{above}(u, v) \rightarrow \text{on top}(u, z) \text{ above}(z, v)$ ;

To answer, consider the question:

$\text{above}(c, a)$ ;

and analyze the answers given by LP in each case.

b) Give, as a tree, the trace of the execution of LP with the program:

1)  $P(a, b) \rightarrow$ ;

```

procedure LP;
input:
A list  $\langle C_1, \dots, C_r \rangle$  of non-negative Horn clause (the program):
 $C_i : A_i \rightarrow B_1^i, \dots, B_{n_i}^i$  ( $n \geq 0; 1 \leq i \leq r$ )
A negative clause (the question):
 $P_1, \dots, P_s$  ( $s \geq 1$ )
output: (if the procedure halts)
 $\square$  or 'no'
begin
 $Res \leftarrow \langle P_1, \dots, P_s \rangle;$ 
 $X \leftarrow P_1;$ 
    while  $Res \neq \square$  and  $Res \neq \text{'no'}$ 
    do
    if there exists j (take the smallest in the order of untreated clauses)
    such that  $A_j \doteq X$  has a solution  $\sigma$ 
    [We memorize all other possible choices to return to them
    in case of failure AND also of success]
    then  $Res \leftarrow \sigma \langle B_1^j, \dots, B_{n_j}^j \rangle \circ \langle Res - \langle X \rangle \rangle$ 
    %  $\circ$  denotes list concatenation
    % Application of the resolution rule with a given strategy
    % We have a stack: last in first out
     $X \leftarrow$  first literal of  $Res$ 
    else  $Res = no$ 
    enddo
end
    
```

Figure 6.1. Abstract interpreter of language LP

1)  $P(c, a) \rightarrow;$

1)  $P(x, y) \rightarrow P(x, z) \quad P(z, y);$  % transitivity

and the *question*:

$P(u, b);$

The syntax of the question has been adapted to existing software.

The question in (b) (and similarly for (a)) can be interpreted as:

– The problem is to know whether there exist objects in relation via  $P$  with  $b$ ;

or, as we used to do in the analysis of reasonings:

– We negate the conclusion, i.e.  $\neg\exists uP(u, b) \rightsquigarrow \forall u\neg P(u, b)$ , and if the reasoning is correct, we must produce  $\square$ .

REMARK 6.4.– We can imagine (and this has indeed been studied and experimented) an interpreter that uses parallelism to handle non-determinism.

With the program:

$$P(x, y) \rightarrow P(y, x);$$

and the question:

$$P(a, b) \quad Q(c);$$

a sequential interpreter does not find any solution and does not halt.

Whereas an interpreter using an AND parallelism (i.e. searching simultaneously for the solution to all the goals of the question) halts without finding a solution, which is normal because it is impossible to solve  $Q(c)$ .

With the program:

$$P(x, y) \rightarrow P(y, x);$$

$$P(a, b) \rightarrow;$$

and the question:

$$P(a, b);$$

a sequential interpreter does not find any solution and does not halt.

Whereas an interpreter using an OR parallelism (i.e. searching simultaneously for all the solutions to one goal of the question) finds a solution.  $\square$

### 6.3. Logic programming: examples

EXERCISE 6.2.– Assuming that  $\mathbb{N}$  and elementary arithmetic operations are not incorporated into the interpreter, give the logic programs corresponding to:

a) **add**  $(x, y, z): x + y = z$

b) **mult**  $(x, y, z): x \times y = z$

c) **less**  $(x, y): x < y$

d) **divides**  $(x, y): x$  divides  $y$ , with mathematical notations,  $x \mid y$  (examples: divides (3, 15), divides (2, 8)  $\neg$  divides (3, 10))

e) **prime** $(x): x$  is a prime number.  $\square$

EXERCISE 6.3.– Assuming that  $\mathbb{N}$  and elementary arithmetic operations are not incorporated into the interpreter, give a logic program that permits us to define the relation:

$\text{fibonacci}(n, x)$ : the value  $\text{fibonacci}(n)$  is  $x$ . □

DIGRESSION 6.1.– (regular expressions). Let  $\Sigma$  denote an alphabet (i.e. a finite set of symbols).

The set of *regular expressions* (r.e.) on  $\Sigma$  is defined as the smallest set such that:

i)  $\emptyset$  is a r.e.;

ii) the set  $\{\epsilon\}$  is a r.e.  $\% \epsilon$  denotes the empty string;

iii) if  $a \in \Sigma$  then  $\{a\}$  is a r.e.

iv) if  $R$  and  $S$  are r.e., then  $R \cup S$ ,  $(R + S)$ ,  $RS$ , and  $R^*$  are r.e. ( $R^*$  is called the *Kleene closure*)

where:

$$RS = \{xy \mid x \in R, y \in S\}$$

$$R^0 = \{\epsilon\}$$

$$R^i = RR^{i-1}$$

$$R^* = \bigcup_{i=0}^{\infty} R^i$$

$$R^+ = \bigcup_{i=1}^{\infty} R^i$$

By convention, for singletons, we identify  $\{a\}$  with  $a$ . □

EXERCISE 6.4.– Given the following logic program:

1)  $\text{fact}(0, 1) \rightarrow$ ;

- 2)  $fact(n + 1, (n + 1) \times y) \rightarrow fact(n, y)$ ;  
 3)  $fact(u, v)$

where  $\alpha, \beta, \gamma, \delta$  below denote the following substitutions (obtained by application of the resolution rule):

$$(1, 1) - (3, 1) \alpha = \{u \leftarrow 0, v \leftarrow 1\}$$

$$(1, 1) - (2, 2) \beta = \{n \leftarrow 0, y \leftarrow 1\}$$

$(2, 1) - (2, 2) \gamma = \{n' \leftarrow n + 1, y' \leftarrow (n + 1) \times y\}$  %  $n'$  and  $y'$  take the renaming into account

$$(2, 1) - (3, 1) \delta = \{u \leftarrow n + 1, v \leftarrow (n + 1) \times y\}$$

Characterize all possible runs of the program using a r.e. that expresses the sequence of substitutions that are applied.  $\square$

EXERCISE 6.5.– In a commonly used LP language, lists are represented as:

$$[a, b, c, \dots, d]$$

$[a_1, a_2, \dots, a_n \mid X]$ :  $a_i$  ( $1 \leq i \leq n$ ) is the  $i$ th element in the list, and  $X$  is what remains.

Define the following predicates (relations) in this language:

- a)  $append(x, y, z)$ : List  $z$  is the concatenation of lists  $x$  and  $y$ ;  
 b)  $reverse(x, y)$ : List  $y$  is list  $x$  reversed;  
 c)  $palindrome(x)$ : List  $x$  is a palindrome;  
 d)  $member(x, y)$ :  $x$  is a member of list  $y$ ;  
 e)  $subset(x, y)$ :  $x$  is a subset of  $y$  (where sets are represented by lists);  
 f)  $consec(u, v, x)$ : elements  $u$  and  $v$  are consecutive in list  $x$ .  $\square$

Can we do classical algorithmic in LP?

The answer is yes, the key to doing so: go back to the specification.

EXAMPLE 6.4.– (syntactic analysis). Give a logic program that recognizes the words of the grammar whose production rules are:

$$S \longrightarrow c$$

$$S \longrightarrow aSb$$

% This rule *does not specify* how to associate symbols in the string

We write  $f(a, b) : a \bullet b$  % string concatenation

Word(c);

(\* Word(f(a,f(x,b)))  $\longrightarrow$  Word(x);

If we want to know whether the word *aacbb*, which is written as:

$$a \bullet ((a \bullet (c \bullet b)) \bullet b)$$

belongs to the language generated by the grammar, we ask the question

Word(f(a,f(f(a,f(c,b)),b)));

that the interpreter will transform into:

Word(f(a,f(c,b)));

Word(c);

and answer:

yes

Note that if we ask the following question:

Word(f(f(a,c),b)), i.e. for the word *acb* written:  $(a \bullet c) \bullet b$

the answer will be

no!

This answer may seem surprising at first, but is “normal” as no particular property has been assumed on the functions denoted by functional symbols, in this case associativity (which is known to hold for  $\bullet$ , see unification algorithm, section 4.2).

If we wanted a correct answer to this second question, we would have had to write clause (\*) as follows:

```
Word(f(f(a,x),b)) → Word(x);
```

but in this case, the first question would have resulted in a surprising answer!

Another logic recognition program that avoids these problems would be:

```
Word(c);
```

```
Word(z) → append([a|x], [b], z) Word(x);
```

In example 9.22, we give another version that is very similar, but uses string constraints, for which there is no problem to be handled by the user.  $\square$

REMARK 6.5.– If we had had a unification algorithm capable of handling *associative* functions, i.e. functions  $f$  with the property:

$$(\forall x \forall y \forall z) f(x(f(y, z)) = f(f(x, y), z)$$

We would not have obtained two different answers, depending on the way the symbols in the word were associated.  $\square$

EXAMPLE 6.5.– (sorting).

```
sort(x,y) → perm(x,y) ord(y);
```

```
% sort(x,y): list y is the sorted list obtained from list x.
```

```
% perm(x,y): list y is a permutation of list x.
```

```
% ord(y): list y is an ordered list.
```

```
perm([],[]) →;
```

```
perm(z,[x|y]) → delete(x,z,u) perm(u,y);
```

```
% delete(x,z,u): u is the list obtained after deleting element x from list z.
```

```
delete(x,[x|y],y) →;
```

```
delete(x,[y|z],[y|u]) → delete(x,z,u);
```

```
ord([]) →;
```

```
ord([x]) →;
```

```
ord([x,y|z]) → x <= y ord([y|z]);  $\square$ 
```



Other versions:

EXAMPLE 6.6.–

```

sort(x,y) → perm(x,y) ord(y);
perm([],[]) →;
perm([e|x],z) → insert(e,y,z) perm(x,y);
% insert(e,y,z): list z is obtained by inserting element e in list y
insert(e,x,[e|x]) →;
insert(e,[u|x],[u|y]) → insert(e,x,y);

```

Another one:

EXAMPLE 6.7.–

```

sort([],[]) →;
sort([u|x],y) → insert(u,z,y) sort(x,z);
insert(x,[],[x]) →;
insert(x,[u|y],[u|z]) → u<x insert(x,y,z);
insert(x,[u|y],[x,u|y]) → x<=u;
% [x,u|y] is equivalent to [x|u|y]

```

REMARK 6.6.– This last logic program corresponds to the insertion sort: we remove an element from the list (here the first one), sort the rest of the list and reinsert the element that was removed, while respecting the order.  $\square$

EXERCISE 6.6.– Give the logic program corresponding to the quick-sort algorithm: we select an element  $e$  in a list and divide the list into two sublists, the elements that are smaller than  $e$  and those that are greater than  $e$ , then  $e$  is prefixed by the sorted list of elements that are smaller than  $e$  and suffixed by those that are greater than  $e$ .  $\square$

EXERCISE 6.7.– Give the logic program corresponding to the bubble sort algorithm: we test if two adjacent elements are not in the correct order. If so, they are swapped. We repeat the operation until no additional swap is necessary.  $\square$

EXAMPLE 6.8.– (automaton). We want to define a program that recognizes the words that are accepted by an automaton.

An automaton is defined by a 5-tuple:

$$\langle Q, \Sigma, \delta, q_0, F \rangle$$

where:

$Q$ : set of states;

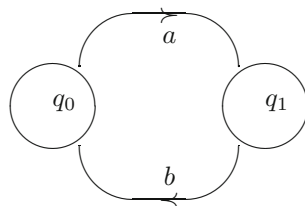
$\Sigma$ : vocabulary;

$\delta: Q \times \Sigma \rightarrow Q$ ;

$q_0$ : initial state;

$F$ : set of final states.

For the following automaton (that accepts  $(ab)^*$ ), with initial state  $q_0$  and final state  $q_1$



where strings are represented as lists, the program is ( $q, x, y, z, u$  denote variables):

- 1) `initial(q0) → ;`
- 2) `final(q1) → ;`
- 3) `delta(q0,a,q1) → ;`
- 4) `delta(q1,b,q0) → ;`
- 5) `accepted-string(x) → initial(q) accept(q,x);`
- 6) `accept(q, []) → final(q);`
- 7) `accept(q, [x|y]) → delta(q,x,q1) accept-next(q1,y);`
- 8) `accept-next(q1, [z|u]) → delta(q1,z,q0) accept(q0,u);`

a question could be, for example:

`accepted-string([a,b,a,b,a,b]);`

□

**6.3.1. Acting on the execution control: cut “/”**

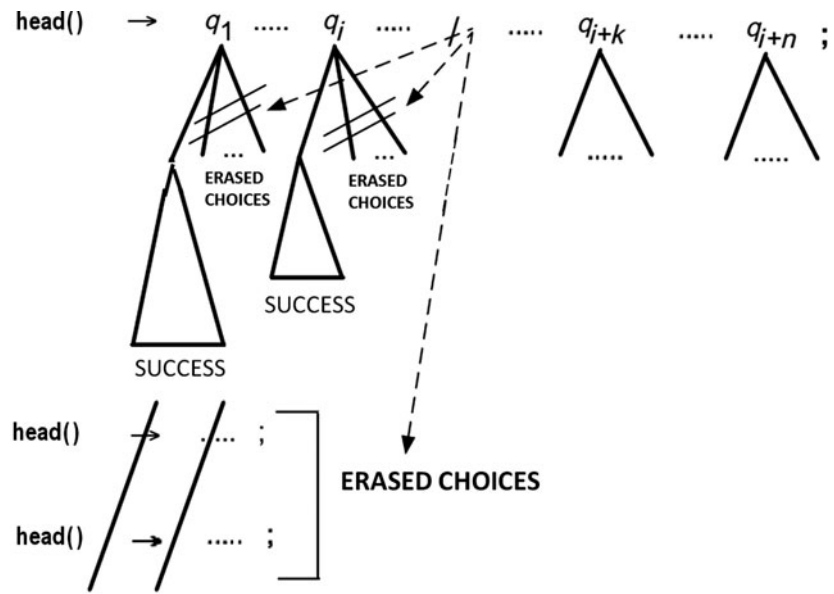
*In principle*, it is not necessary in logic programming to know *how* a program is executed. However, in reality, it is necessary to control executions, which is why imperative characteristics have been introduced into this declarative framework. The cut rule is not very elegant, but is extremely useful, although it must be used very carefully.

The definition given in the use manual is the following:

The “/” is a parasite that can only occur among the terms that make up the right-hand side of a rule (in particular, in the question). The choices that remain to be examined and that the deletion (it is always deleted) of “/” removes are as follows:

- the other rules that have the same head as that of the rule where “/” occurs.
- the other rules that could have been used to erase the terms occurring between the beginning of the tail and “/”.

If we represent the execution tree:



- / : goal that always succeeds... but with...
- : a side-effect, after the success of ‘/’

EXAMPLE 6.9.– (from the manual). The program:

```

colour(red) →;
colour(blue) →;
size(big) →;
size(small) →;
choice1([x,y]) → colour(x) size(y);
choice1(«that's all») →;
choice2([x,y]) → / colour(x) size(y);
choice2(«that's all») →;
choice3([x,y]) → colour(x) / size(y);
choice3(«that's all») →;
choice4([x,y]) → colour(x) size(y) /;
choice4(«that's all») →;

```

gives the following answers to the asked questions:

```

choice1(u);
{u= [red,big]}
{u= [red,small]}
{u= [blue,big]}
{u= [blue,small]}
{u= «that's all»}
choice2(u);
{u= [red,big]}
{u= [red,small]}
{u= [blue,big]}
{u= [blue,small]}
choice3(u);
{u= [red,big]}
{u= [red,small]}
choice4(u);
{u= [red,big]}
choice1(u) /;
{u= [red,big]}.

```

□

It is not hard to show how useful the cut is; some examples are given below:

6.3.1.1. *Translation of imperative structures*

For several reasons (for example, because they are more natural, such as in the case of inputs/outputs), we can translate well-known control structures:

if P then Q else R:

if-P-then-Q-else-R  $\rightarrow$  P/Q;

if-P-then-Q-else-R  $\rightarrow$  R;

If P succeeds (is erased), the cut erases the choice of the second rule and problem Q is considered. If P cannot be erased, problem R will be considered.

while  $\neg$  R do Q:

P(X)  $\rightarrow$  R(X) /;

P(X)  $\rightarrow$  Q(X,Y) P(Y);

EXERCISE 6.8.– Give the execution trace of the program:

- 1) R(4)  $\rightarrow$ ;
- 2) 2. Q(1,2)  $\rightarrow$ ;
- 3) 3. Q(2,3)  $\rightarrow$ ;
- 4) 4. Q(3,4)  $\rightarrow$ ;
- 5) 5. P(X)  $\rightarrow$  R(X) /;
- 6) 6. P(X)  $\rightarrow$  Q(X,Y) P(Y);

with the question:

P(1);

□

Combination with the predefined predicate `repeat`.

Here is a predicate that may seem strange if we only think of using it by itself:

```
repeat  $\rightarrow$ ;
```

```
repeat  $\rightarrow$  repeat;
```

...but here is an example that makes it less strange:

```
read-spaces (' ', c)  $\rightarrow$  repeat in-char(c) dif(c, ' ');
```

```
read-spaces (c,c)  $\rightarrow$ ;
```

In this program, `in-char( $\tau$ )`: read a character from the input stream.

EXAMPLE 6.10.– (set operations). We represent sets as lists.

```

member(x,y) → append(u,[x|z],y);

other program
member(x,[x|y]) →;
member(x,[z|y]) → member(x,y);
non-member(x,[]) →;
non-member(t,[u|x]) → dif(t,u) non-member(t,x);
subset([],x) →;
subset([z|x],y) → member(z,y) subset(x,y);
inter([],x,[]) →;
inter([x|u],y,[x|z]) → member(x,y) / inter(u,y,z);
inter([x|u],y,z) → inter(u,y,z);
union([],x,x) →;
union([x|u],y,z) → member(x,y) / union(u,y,z);
union([x|u],y,[x|z]) → union(u,y,z);

```

### 6.3.2. Negation as failure (NAF)

The negation problem is a very delicate problem (including from a philosophical point of view). For example, what can we say about a goal (predicate, problem) that we know we will not be able to prove with the data at our disposal? A solution would be to say “we do not know”.

Another solution is to use the closed world assumption: if we are sure that we cannot prove  $P$  (which prevents infinite searches), we conclude  $\neg P$ .

This second possibility is the possibility that was chosen. It corresponds to the addition of the rule:

$$NAF : \frac{no \vdash P}{\vdash \neg P}$$

(Here, “no  $\vdash P$ ” means: after an exhaustive search and halting).

Usual implementation of negation in Prolog:

```

not(Z) → Z / fail;
not(Z) →;

```

where  $Z$  is a predicate variable, meaning that it is mapped to predicates and

`fail`: predicate (goal) that always fails (that cannot be erased).

Using `not`, we can give another definition of `if...then...else`:

`if-P-then-Q-else-R`  $\rightarrow$  `P Q`; `if-P-then-Q-else-R`  $\rightarrow$  `not(P) R`;

The choice that was made to treat negation may seem natural, but it leads to several problems, as illustrated in the following examples and exercises.

EXAMPLE 6.11.– The program:

`p`  $\rightarrow$  `not(q)` % equivalent to `p`  $\vee$  `q` (if `not` is identified with  $\neg$ )

with the question

`p`;

succeeds, but this means that from `p`  $\vee$  `q`, we can deduce `p`! □

EXAMPLE 6.12.–

```
1) correctalg(algo1)  $\rightarrow$ ;
2) correctalg(algo2)  $\rightarrow$ ;
3) correctalg(algo3)  $\rightarrow$ ;
4) correctalg(algo4)  $\rightarrow$ ;
5) costlyalg(algo1)  $\rightarrow$ ;
6) costlyalg(algo4)  $\rightarrow$ ;
7) reasonablealg(X)  $\rightarrow$  not(costlyalg(X));
```

with the question:

```
correctalg(X) reasonablealg(X);
```

the answer is:

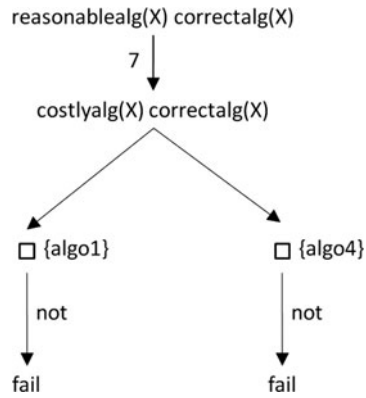
```
{X = algo2 }
```

```
{X = algo3 }
```

... but with question:

```
reasonablealg(X) correctalg(X);
```

the answer is no, as shown by the execution tree:



□

EXERCISE 6.9.– Give the answers to the following questions:

a)

- 1) `man(adam) →;`
  - 2) `woman(x) → not(man(x));`
  - q1) `woman(adam); % question 1`
  - q2) `woman(eve); % question 2`
  - q3) `woman(x) eq(x,eve); % question 3`
- `% see definition of eq in exercise 6.10 (b).`

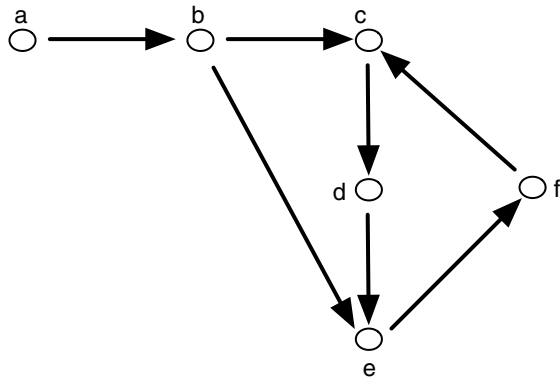
b)

- 1) `even(0) →;`
- 2) `even(s(s(x))) → even(x);`
- 3) `odd(s(0)) →;`
- 4) `odd(s(s(x))) → not(even(x));`
- q1) `even(y); % question 1`
- q2) `odd(y); % question 2`

□

EXAMPLE 6.13.– (beware of non-termination!). We want to write a program that detects whether there is a path from one vertex to another in the graph below (compare with example 6.1).





The program that describes the graph and gives the definition of a path is the following:

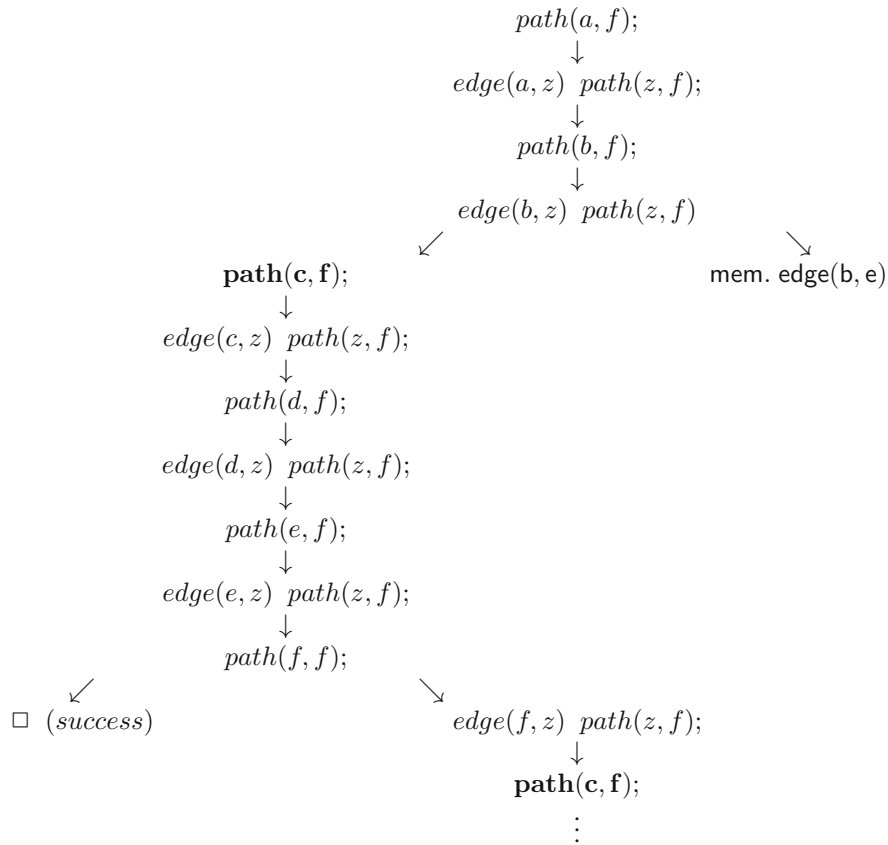
```

edge(a,b) → ;
edge(b,c) → ;
edge(c,d) → ;
edge(b,e) → ;
edge(d,e) → ;
edge(e,f) → ;
edge(f,c) → ;
path(x,x) → ;
path(x,y) → edge(x,z) path(z,y);
  
```

It seems obvious that this program is correct, meaning that it describes the world and the desired relations between objects in this world. It is correct, but... let us see what the interpreter LP does (which, as long as there are potential solutions, keeps trying to find them).

We ask the question:

```
path(a,f);
```



A solution is to replace the last two clauses by the clauses:

`path(x,x,t) → ; % t: list of visited nodes`

`path(x,y,t) → edge(x,z) not(member(z,t)) path(z,y,[z|t]);`

and the question would then be:

`path(a,f,[]);` □

EXERCISE 6.10.– Simulate the interpreter to show that the answers are indeed those given below.

a) Consider the program:

1) `r(aa) →;`

2)  $q(X) \rightarrow \text{not}(r(X));$

3)  $p(X) \rightarrow \text{not}(q(X));$

the question:

$p(aa);$

succeeds

the question:

$p(X)$

succeeds and gives as a result the empty substitution (when a goal fails, substitutions are not memorized, which is a reasonable choice).

b) There exists in Prolog a predicate  $\text{eq}(T1, T2)$ , that can be evaluated, and has the following effect when  $T1$  and  $T2$  are variables, say  $X$  and  $Y$

$\text{eq}(X, Y)$ : if  $X$  and  $Y$  have been linked to the same term, **true** else if  $Y$  (respectively,  $X$ ) is not yet linked, then it is linked to the same term as  $X$  (respectively,  $Y$ ) else **false**.

(This does not correspond to usual equality, see section 9.1).

Consider the program:

1)  $\text{eq}(X, X) \rightarrow;$

2)  $p \rightarrow \text{not}(\text{eq}(X, 1)) \text{eq}(X, 2);$

the question:

$p;$

fails.

It should succeed (with  $X=2$ )

the question:

$\text{not}(p);$

succeeds. □

EXERCISE 6.11.– Give a logic program for the relation  $\text{merge}(X, Y, Z)$ :  $Z$  is a sorted list of integers obtained by merging the two sorted lists of integers  $X$  and  $Y$ . □

6.3.2.1. *Some remarks about the strategy used by LP and negation as failure*

A program containing clauses such as (\*) below causes problem at the execution of the program, because it artificially introduces a possibility of non-termination.

The recommended solution is quite natural.

EXAMPLE 6.14.– a) If a program contains the clauses:

(\*)  $cc(X,Y) \rightarrow cc(Y,X)$  ;

$cc(s,t) \rightarrow \text{clause-queue}$  ;

with  $s,t$ : terms

replace (\*) by:

$cc(t,s) \rightarrow \text{clause-queue}$  ;

b) If a program contains the clauses:

(\*)  $\text{circ}(Y,Z,X) \rightarrow \text{circ}(X,Y,Z)$

$\text{circ}(s,t,u) \rightarrow \text{clause-queue}$  ;

with  $s,t,u$ : terms

replace (\*) by:

$\text{circ}(u,s,t) \rightarrow \text{clause-queue}$  ;

$\text{circ}(t,u,s) \rightarrow \text{clause-queue}$  ;

□

EXAMPLE 6.15.– (evaluation of formulas). We can sometimes use NAF to evaluate a logical formula in a database that defines the predicates occurring in the formula.

The (very simple) idea is that *if we identify not with  $\neg$  and the question  $A_1 A_2 \dots A_n \text{ not } (B)$*

– fails then the formula  $A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow B$  is evaluated to **T**;

– succeeds then the formula  $A_1 \wedge A_2 \wedge \dots \wedge A_n \Rightarrow B$  is evaluated to **F**.

For example, if in the database:

1)  $p(1) \rightarrow$  ;            4)  $q(1,2) \rightarrow$  ;

2)  $p(2) \rightarrow$  ;            5)  $q(2,1) \rightarrow$  ;

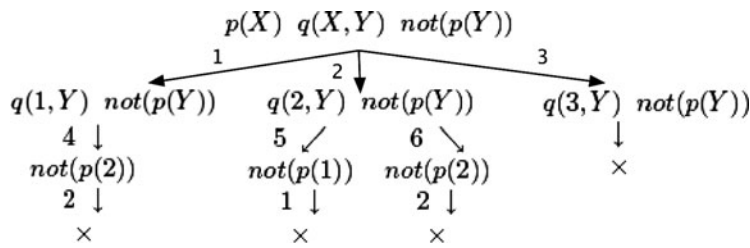
3)  $p(3) \rightarrow ;$       6)  $q(2,2) \rightarrow ;$

we want to evaluate the formulas:

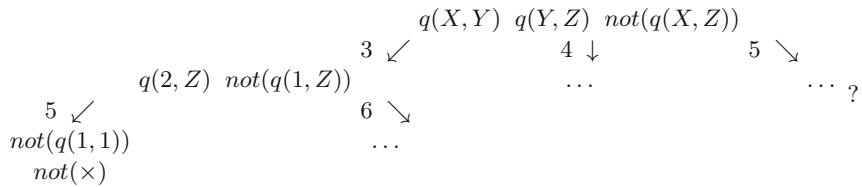
a)  $\forall x \forall y. P(x) \wedge Q(x, y) \Rightarrow P(y)$

and:

b)  $\forall x \forall y \forall z. Q(x, y) \wedge Q(y, z) \Rightarrow Q(x, z)$



Hence, (a) is a valid formula in this world.



Formula (b) is therefore not valid. We could retrieve the substitution:

$\{X = 1, Y = 2, Z = 1\}$

that corresponds to the counterexample:

$Q(1, 2) \wedge Q(2, 1) \Rightarrow Q(1, 1).$  □

6.3.2.2. Can we simply deduce instead of using NAF?

EXAMPLE 6.16.– (completion (Clark)). The program:

`math-class(E106) → ;`

`math-class(E108) → ;`

will answer no to the question

`math-class(E201);`

Using the *NAF* rule.

To *deduce* (without any other rule than resolution) this conclusion, we must explicit what is assumed by *NAF*:

$$\forall x. \text{math-class}(x) \Rightarrow (x = E106) \vee (x = E108)$$

and:

$$E106 \neq E108$$

$$E106 \neq E201$$

$$E108 \neq E201$$

$$E108 \neq E106$$

$$E201 \neq E106$$

$$E201 \neq E108$$

The program in clausal form (non-Horn) would then be:

$$1) \text{math-class}(E106)$$

$$2) \text{math-class}(E108)$$

$$3) \neg \text{math-class}(x) \vee (x = E106) \vee (x = E108)$$

% Clause 3 is not a Horn clause

$$4) \neg(E106 = E108)$$

$$5) \neg(E106 = E201)$$

$$6) \neg(E108 = E201)$$

$$7) \neg(E108 = E106)$$

$$8) \neg(E201 = E106)$$

$$9) \neg(E201 = E108)$$

And we would ask the question

$$\neg \text{math-class}(E201);$$

we negate the conclusion to obtain:

$$10) \text{math-class}(E201)$$

and by resolution:

$$11) E201 = E106 \vee E201 = E108 \quad (3, 1) - (10, 1)$$

$$12) E_{201} = E_{108} \quad (11, 1) - (8, 1)$$

$$13) \square \quad (12, 1) - (9, 1). \quad \square$$

REMARK 6.7.– (on the utility of studying logic programming). Together with the fact that there are many direct applications that are easy to imagine, starting with examples that have already been seen, the study of the principles of LP is very useful as a first step toward the study of ontology languages (see section 1.2 and digression 8.1), in particular, the so-called *description logics* in which knowledge bases are made of *assertions* (i.e. properties of individuals) and *terminological axioms* (i.e. complex descriptions). The descriptions are specified with unary and binary predicates.

The analogy to LP is obvious.

In the domain of databases, there exists a family of languages (Datalog) based on rules that are clauses (with some restrictions) of FOL. They represent a very important domain of study and can be considered as a “natural” extension of LP.

Datalog perfectly illustrates the deep relationship that exists between logic and computer science.  $\square$

#### 6.4. Computability and Horn clauses

At this point, the reader who has already written programs in so-called *functional* languages (Lisp, Scheme, etc.) has probably noticed the similarities between programs in these languages (or simply between the definitions of functions with equations, such as *factorial*, *fibonacci*, etc.) and logic programs.

Furthermore, when presented with a new language, say,  $\mathcal{L}$ , it is natural to wonder what its expressive (and computational) power is, in other words, if it enables us to define (and compute) all computable functions.

One way of answering this question is to show that  $\mathcal{L}$  can be used to encode a Turing machine or Markov algorithms, etc. or that it can be used to encode operations that permit us to capture all computable functions (minimization, etc.).

We will choose this way of proceeding, as it will enable us to show the computational power of Horn clauses, and at the same time relate functional programming and LP (also called *predicative* programming or *relational* programming).

We first briefly recall the definition of a function that is *finitely definable by equations*.

We will add to  $\mathcal{F}$  (see signatures in definition 5.1) the constant  $\mathbf{0}$  and the unary function  $\mathbf{s}$  (for successor), and we shall use the notion of terms (see definition 4.1) on the modified signature, which will be sufficient to suit our needs.

Natural numbers (which are necessary to define functions  $\mathbb{N}^k \rightarrow \mathbb{N}$ ) will be denoted by  $s^{(n)}(\mathbf{0})$ .

The variable domain will be the terms of the form  $s^{(n)}(\mathbf{0})$  ( $0 \leq n$ ).

The equations that will occur in the definitions will be term equations, defined on the new signature, with all their variables universally quantified. We shall impose that the subterms  $t_i$  ( $1 \leq i \leq n_i$ ) of one of the terms in the equations, say  $f_i^{n_i}(t_1, \dots, t_{n_i})$ , only contain the functional symbols  $\mathbf{0}$  and  $s$ . The (unique) function conventionally denoted by  $f_i$  is the function defined by the equational system.

An equational system is a finite set of equations.

We denote the equational systems (and the equations) by  $\mathcal{E}(f_1, \dots, f_n ; \bar{x})$  (for the sake of readability, we omit the exponent representing the arity; this does not incur any confusion).

$f_1, \dots, f_n$  and  $\bar{x}$ , respectively, represent the set of function symbols and the set of variables that occur in the equations.

**DEFINITION 6.1.**– (functions definable by equations). A function  $f_i$  is finitely definable by an equational system iff there exists an equational system  $\mathcal{E}$  such that:

–  $\exists f_1 \dots \exists f_n \forall \bar{x} \mathcal{E}(f_1, \dots, f_n ; \bar{x})$  (meaning that  $f_1, \dots, f_n$  satisfy all the equations in the system);

– for any set of variables  $\bar{y}$ , there exists a finite set  $\bar{z}_1, \dots, \bar{z}_k$  such that:  
 $\mathcal{E}(f_1, \dots, f_n ; \bar{z}_1) \wedge \dots \wedge \mathcal{E}(f_1, \dots, f_n ; \bar{z}_k)$  and this set uniquely determines  $f_i(\bar{y})$ .

The following theorem (the proof of which can be found in textbooks on computability) characterizes the set of computable functions by functions defined by equational systems.

**THEOREM 6.1.**– Every recursive function is finitely definable by an equational system.

**EXAMPLE 6.17.**– (length of a list). We add to the signature (i.e. to  $\mathcal{F}$ ) the function symbols *nil*, *length*, *cons*, *add*.

$$\text{length}(\text{nil}) = 0$$

$$\text{length}(\text{cons}(u, v)) = \text{add}(\text{length}(v), 1) \quad \% \text{ we write 1 instead of } s(\mathbf{0}). \quad \square$$



The relationship between equational definitions and computability is established by theorem 6.1, but what is the relationship between these definitions and Horn clauses and their expressive power?

The first key remark is that an equation can be expressed as an implication, in the form of a Horn clause:

**Rule I:**

$$A_1 \wedge \dots \wedge A_n \Rightarrow B$$

with  $A_i$  ( $1 \leq i \leq n$ ) and  $B$  equational literals (i.e. literals for which the predicate symbol is  $=$ ), with arguments that are terms of depth 1 or 2 (or 0 or 1 depending on the conventions, see definition 4.2). These terms are called flat terms, meaning that they are of the form:

$$f_i^{(n)}(x_1, \dots, x_n) = y$$

where

$x_i, y$ : variables (universally quantified) or constants.

The second key remark is that, to obtain flat terms it suffices to name them.

We show how rule I can be used on example 6.17.

1)  $length(nil) = 0$  % as nil is a constant, the equation has the desired form

$$2) \underbrace{length(cons(u, v))}_x = \underbrace{add(length(v), 1)}_y = \underbrace{\hspace{10em}}_z$$

Hence, 2. can be rewritten as:

$$2') length(v) = y \wedge add(y, 1) = z \wedge cons(u, v) = x \Rightarrow length(x) = z$$

To get to a logic without equality, we use (the only if part of)

**Rule II:**

$$f(x) = y \text{ iff } F(x, y)$$

For the example on the length of a list, this yields:

EXAMPLE 6.18.– (length of a list, Horn clauses without  $=$ ). (We use the syntax from section 6.2)

$\text{Length}(\text{nil}, 0) \rightarrow;$

$\text{Length}(u \mid v, z) \rightarrow \text{Length}(v, y) \text{Add}(y, 1, z);$  % where  $u \mid v$  is another way of writing  $\text{cons}(u, v)$

In a constraint LP language (see section 9.2), the second clause could be written:

$\text{Length}(u \mid v, z) \rightarrow \text{Length}(v, y) \{z = y + 1\};$  □

*Conclusion:* rules I and II provide a mechanical way of getting from equational definitions to Horn clauses without equality, which proves that the latter permit us to compute all computable functions.

REMARK 6.8.– The first proof of computability with Horn clauses was produced in 1975–1976. The proof technique consisted of showing that a Turing machine could be encoded with Horn clauses.

More precisely, it was proved that if a function is computable by a Turing machine, then it is computable with Horn clauses containing at most one negative literal (and at most one functional symbol).

An immediate corollary is that the class of Horn clauses is *undecidable*. □

## Chapter 7

# Artificial Intelligence

*Intelligence is by essence unintelligible.*

*David Hume (1711–1776)*

Although we often believe that the philosopher is right, we cannot deny that scientific progress, in particular in biology, has permitted us to cast a light on the following problems: studying intelligence, its partial simulation on machines, what can be formalized (with the current state of formal tools), appreciating its limits, better identify where the problems are important, and so on.

Furthermore, note that from a *practical* point of view, we *need* more and more intelligent tools (whatever the reasonable and perhaps informal characterization of “intelligence” we adopt).

### **7.1. Intelligent systems: AI**

To begin to grasp the topic, we recall the etymology of (natural) intelligence<sup>1</sup>:

in French:

Intelligence: 12th Century “understanding”, 15th Century “communication between people who understand each other”.

---

<sup>1</sup> The term “intelligent” is currently used, often in a glamorous way, on anything that has more or less surprising properties (for example, in nanotechnology, there are “intelligent materials”).

in English:

**Intelligent:** 16th Century, from the Latin words *intellegere*, *ligere* lit. *choose among, formed on INTER + legere gather, choose.*

In other words, *choose among* (which implicitly seems to admit the importance of *handling non-determinism*).

As far as its study and modeling is concerned, for a long time, mathematics has been used to study problems that arise in natural science, in particular on the behavior of the brain and the nervous system. Conversely, important developments in different topics of mathematics have been motivated by these problems. All this happened before there was even a research domain called AI.

DIGRESSION 7.1.— Ramón Llull (c. 1235–1315) and G. Leibniz (1646–1716) are often cited as pioneers of the “AI project”, i.e. of the belief that every thought (the essential human characteristic) can be formalized. This project became more concrete, at least partially, with the development of computers, and its huge influence on science and society in general.

Ramón Llull designed a reasoning machine (more precisely a deduction machine) with which he wanted to encode (in a combinatorial way) knowledge of creation in a universal language combining base symbols.

G. Leibniz believed that if we could produce a list of basic human thoughts (i.e. words denoting simple ideas), it would be possible to produce mechanically (by combination) all complex ideas. (This should be compared with the concept of a formal system.)

This reductionism seems to have a very long history. Some religions (in their esoteric approach) postulated on the existence of absolute ideas and of a mathematic (algebra) of ideas. □

The models that were used were constructed based on neural models. Some chemical, electrical, and mechanical aspects were taken into account and differential equations were used as tools of modeling and analysis.

This tradition continues in a part of current AI, another larger part inherited from the discrete modeling approach (two states, 0 and 1), which originated with the work of McCulloch and Pitts, Turing, and others.

Of course, the study of other models, thanks to other tools, is possible (and likely) in the future.

Recently, some research has focussed on the cerebral bases of mathematical activity, for example, on the cerebral activity that corresponds to the understanding of numbers and calculus.

The importance of these studies is reflected in the creation at the *Collège de France* of a chair of “experimental cognitive psychology”.

How can AI be defined? For example, consider the following definition by Minsky, one of the pioneers of the domain.

Artificial Intelligence is the science of making machines do things that would require intelligence if done by men.

Is this definition really satisfactory? Consider motion for example, as well as the laws that characterize it. Motion can be defined independently from the objects that move... but intelligence has probably never been defined independently from the beings that we consider as intelligent.

If in Minsky’s definition we replace “intelligence” by “kinematics”:

Kinematics is the science of making machines do things that would require motion [if done by cars].

[]: redundant here but not in the previous definition.

If we say that intelligence (or that intelligent behavior) is an exclusive property of living beings (or of the animal kingdom), the problem is somehow solved: a computer system cannot exhibit an intelligent behavior, except if we believe a computer can be alive. We have then replaced the problem of defining intelligence by the problem of defining living beings.

At least since cybernetics (see section 7.5.2), connections have been made with life (in particular, with the way the brain works): *self-reproducing automata* (artificial), neural networks, etc.

REMARK 7.1.– The following definition is sometimes given to artificial life (a domain related to AI):

Artificial life denotes the study of artificial systems that exhibit a behavior that is characteristic to natural living systems.

This definition is similar to Minsky’s on (which would be a particular instance of artificial life), and in our opinion, is not satisfactory either.

To be an interesting definition, it presupposes that there is a decidable way of qualifying all the characteristics related to a living system, or at least all those that are inherent to the *specific distinction* between living and non-living systems.

The two following definitions go in the same direction.

The definition of life:

Life is a self-maintained chemical system that is subject to Darwinian evolution.

and the definition of living organism

A living organism is a chemical system that is capable of regenerating its own constituents, and exchanges matter and energy with its environment; this system is capable of reproducing itself in an imperfect manner, generating slightly different replications of itself, possibly better adapted to the environment.

What do we do with suffering, pleasure, emotions, anguish, feelings, etc. that seem to interact with intelligence?

There seems to be a consensus between researchers on how difficult it is to characterize intelligence, one of the consequences of the evolution of life.

A key concept here is another concept that we shall study: the concept of *explanation*. Is a black box that to some inputs associates outputs that resemble what happens in an organism an explanation of the behavior of the organism? We should be very careful, because, for example, in a domain we know better, the way programs play chess, prove theorems, suggest clauses that explain data... is probably not the same as the way a human would proceed.

In any case, it is worth mentioning that it is generally not wise to identify the dreams of some researches (no matter how brilliant they may be) with reality. For example, two important names in AI predicted in 1958 that as soon as 1970, computers would be capable of composing classical music, writing masterpieces, discovering theorems, playing chess, understanding and translating languages, etc.

Of all these predictions, we can say that that until now, and although considerable progress has been made, the only one that was completely realized (in 2006) is the one on the game of chess (the program Deep Fritz defeated the world champion in six games, with two victories and four draws).

REMARK 7.2.— In the analysis of the different ways intelligence can be characterized, we frequently forget to mention a meaningful fact. Radio-astronomers who are searching for proofs of *artificial life* in the universe try to detect *non-random* signals coming from outer space. Producing signals *that respect laws* should therefore be a (sufficient) condition to characterize intelligence.

REMARK 7.3.– In another important activity of human beings: art, researchers (most of them in neurobiology) are trying to “extract” general laws on beauty (i.e. what leads us to qualifying objects, ideas, etc. as “beautiful”).

This domain of study could be named “artificial art” and in our opinion, its importance must not be underestimated. After all, three centuries after his death, Mozart is still considered a genius, whereas generations of scientists have disappeared without a trace. □

## 7.2. What approaches to study AI?

We provide three possible approaches (others can of course be imagined).

- 1) Try to define intelligence independently from humans (animals).
- 2) Replace AI by an expression such as “design of assistant for intelligent tasks [or requiring manual skills. . .]”
- 3) Try to design systems that mimic capabilities that psychology, history, etc. consider as intelligent in a human being.

## 7.3. Toward an operational definition of intelligence

In physics, when a concept is defined by specifying what operations are necessary to measure the terms that occur in the definition, we say that the concept has an operational definition.

In 1950, Alan Turing proposed in his paper “Computing Machinery and Intelligence” an imitation game now known as the Turing test<sup>2</sup>. It is presented in many different ways in the literature, and these presentations do not always correspond to what is explicitly said in the aforementioned paper, and probably not to what Turing thought on the topic. The most popular version seems to be: *A system (machine) that makes you believe you have communicated (or interacted) with a human being can be considered as intelligent.*

Turing remained very cautious in his paper about the notions of thought, conscience, intelligence, etc.

Turing’s paper begins as follows:

I propose to consider the question “Can machines think?”

Turing replaces this question by another question that is closely related, and that he describes as an *imitation game*.

---

<sup>2</sup> A lot of information, discussions, etc. on this test can be found on the Internet.

### 7.3.1. *The imitation game proposed by Turing*

– Three players: a man ( $A$ ), a woman ( $B$ ), and a questioner ( $C$ ) who can be a man or a woman.

– Rules of the game

1)  $A$  and  $B$  are in the same room.

2)  $C$  is in another room, and cannot see or hear  $A$  or  $B$ . He can only communicate through (typed) written messages.

3)  $C$  gives names to the people in the other room. For example, he says:  $X$  is the one on the left-hand side and  $Y$  is the one on the right-hand side.

4)  $C$  can ask questions to  $A$  and  $B$ .

– Goal for the players

-  $A$ : force  $C$  to make a mistake.

-  $B$ : help  $C$  give the correct answer.

-  $C$ : determine who among  $A$  and  $B$  is the man and who is the woman (by saying, for example,  $X$  is  $A$  and  $Y$  is  $B$ ).

– Example of questions and answers

-  $C$ : Will  $X$  please tell me the length of his or her hair?

-  $X$ : My hair is shingled, and the longest strands are about nine inches long.

Turing suggests to replace the question “Can machines think?” with “What will happen when a machine takes the part of  $A$  in this game?”. Will the questioner make a mistake with the same frequency as when the game is played by a man and a woman? (*Turing test*)

Given what Turing wrote, we can deduce that a program (system, etc.) that could replace  $A$  and fool the questioner with the same frequency as a human should be qualified as intelligent or capable of thought.

Some authors attribute a key position to the Turing test in the definition of AI:

Artificial Intelligence is the enterprise of constructing a physical symbol system that can reliably pass the Turing test.

Turing believed (at least, this is what he wrote) that such programs would exist by the end of the 20th Century.

How far are we from Turing's prediction?

In January 2000, there was a congress on the Turing test and a competition was organized: six programs were enrolled.



**Results:**

The questioners gave 91% correct answers after five minutes, 93% after fifteen minutes. No program was able to fool a human.

REMARK 7.4.— Some authors have proposed extensions of the Turing test to take sensorimotor functionalities into account.

These extensions are far from arbitrary and correspond to human evolution. Speech, which is essential for intelligent activity, lies within Broca's area. The development of these zones is related to the standing posture and to manual activity.

The difficulty in designing a test, such as the Turing test, is clearly illustrated by autism. Art seems to be one of the most high-level manifestations of the human spirit (the word the most frequently used to describe Mozart, for example, is *genius*). There are *autistic* children who have *exceptional* qualities for painting and music, or who have mnemonic, or shape recognition capacities that are well beyond those of "normal" people, even those considered to be intelligent. In the domain of mathematics, there also exist autistic people who are prodigies in calculus.

These exceptional capabilities restricted to a *unique* domain<sup>3</sup> have led researchers to postulate on the existence of a multiplicity of intelligences, controlled by rules that are hard-wired in different neural areas. □

Recently, a logician who was interested in the relationship of logic with other domains (S. Buss) wrote:

I wish to avoid philosophical issues about consciousness, self-awareness and what it means to have a soul, etc. and instead seek a purely operational approach to artificial intelligence. Thus I define artificial intelligence as being constructed systems which can reason and interact both syntactically and semantically. To stress the last word in the last sentence, I mean that a true artificial intelligence system should be able to take the meaning of statements into account, or at least act as if it takes the meaning into account.

#### **7.4. Can we identify human intelligence with mechanical intelligence?**

The aim of this argument is to show that trying to identify intelligence (and understanding) with a sequence of states leads to consequences that cannot reasonably be accepted.

---

<sup>3</sup> There are some rare cases in which several of these capabilities have been known to coexist within the same individual.

The philosophical argument that is attacked is the argument that states that computers running programs can possess mental states and that when we possess the same states, similar programs are executed in our brains.

#### 7.4.1. Chinese room argument

This argument (proposed by Searle in 1980) describes a situation showing that an entity can pass the Turing test without us being able to say that it actually *thinks* or *understands* in the traditional sense.

A person  $P1$  who does not understand written (and spoken) Chinese at all is isolated in a room and can only communicate with the outside world through signs written on pieces of paper. The person has paper, a pencil, and an instruction manual (program) written in his mother tongue.

- He is given pieces of paper on which signs are “scrawled”.
- Using these signs and the instruction manual,  $P1$  writes other signs on pieces of paper.

After some time, the experiment stops. A person  $P2$  (the presence of which is ignored by  $P1$ ) is outside and is the person giving the pieces of paper to  $P1$  and receiving those passed out by  $P1$ .  $P2$  understands Chinese perfectly. The papers that were given to  $P1$  contained a story written in Chinese together with questions on this story. The papers that were received contained answers in Chinese.

For  $P2$ , entity  $P1$  passed the Turing test and should be identified as intelligent (see section 7.3)... but of course,  $P1$  does not understand Chinese. Which means that the Turing test can be passed for a spoken language without even understanding it!

Furthermore, everyone has laughed when reading automatic translations or has already been incapable of understanding (in their own mother tongue) the translations produced by translation software that is available on the Internet.

#### Other formulation

1) Algorithms are independent of the hardware on which they are programmed: in particular, the machine can be a human (here, of course, execution time is not taken into account).

2) We assume that there is a program  $P$  in a room that can produce speeches like someone whose mother tongue is Chinese. The program is supposed to produce speeches that a native Chinese speaker could not distinguish from those produced by a human being: the person listening believes someone Chinese is speaking.

3) By assuming the philosophical thesis mentioned above is correct, any system on which program  $P$  is executed understands Chinese.

4) There remains to imagine that the program was executed by a human being who does not know Chinese at all: the conclusion is that the person still does not know Chinese in the usual sense, and neither does the computer.

This is an argument that shows the limits of the Turing test: someone can give the impression of “understanding” without “understanding” anything at all.

REMARK 7.5.— This argument may seem far-fetched and artificially created to defend a thesis.

Yet, it partially corresponds to what happens for the Etruscan language<sup>4</sup>. The following quote is from a book on the history of the Etruscan language (L.J. Calvet) published in 1996:

Etruscan writing does not pose any difficulty: it is an alphabet inspired from Greek, . . . But if we are fully capable of reading this alphabet, we do not really know what language it was transcribing: we can read aloud texts that we do not understand.

This is the same for the Iberian language (the first people of the Iberian peninsula).

Neuropathology also (more sadly) illustrates that some tasks that may seem intelligent to an observer are far from being intelligent.

Among some of the “autistic geniuses”, there are children who from the age of two are able to read books and newspapers very easily, but without understanding a single word (because they are only able to decode texts from a phonological point of view and this expertise does not encompass meaning). □

It is worth recalling that our intellectual faculties can be affected by emotions and feelings (stress diminishes our capacities in general, a taste for certain topics can make them easier to understand, to discover, to solve related problems, etc.).

When the chess master Garry Kasparov lost against a computer in 1997, observers believed that he had been *emotionally disturbed* and thus, had played poorly<sup>5</sup>.

---

4 Not much is known about the Etruscan civilization: it originated in Italy around 700 BC and disappeared around 350 BC.

5 As a neurobiologist wrote: “Thus, the functioning of the limbic system, which supports emotions, memory, and therefore the cognitive system, depends on a perfect and delicate tuning of different neuromodulators. Too much or not enough chemical activity prevents these systems from functioning normally.”

More recently (2006), the world champion Vladimir Kramnik made an unexplainable mistake and lost a game against the software Deep Fritz, a mistake that “normally” even a beginner should not make. . . but *errare humanum est*.

This relation seems to have been appreciated in the language of a great civilization (the Chinese civilization): the ideogram for think is obtained by merging the one for head with the one for heart.

### 7.5. Some history

Humans have tried for a long time to reproduce or mimic life, and in particular creatures that resemble them.

For example, the mythical *Golem*, man-robot created by magical or artificial means.

Furthermore, in mythical stories, labor has always been a punishment and a yoke that man has always tried to get rid of.

We classify the works in two large categories, that we can imagine as increasingly merging and that we will call Prehistory and History.

#### 7.5.1. Prehistory

Emphasis is put on the reproduction of external properties of living beings: movements, gestures, sounds, etc.

The following facts are part of tradition (hard to verify).

- 5th–4th Century BC: carrier pigeon that could fly (Archytas).
- 4th–3rd Century BC: snail that could crawl (Demetrius).
- 4th Century BC: automatic signal to call Plato’s students to class.
- 3rd Century BC: android (Ptolemy II Philadelphus).
- 1st Century AD: theater shows with automata on the return home of the heroes of the Trojan war (Hero of Alexandria).

The name of this Greek engineer and mathematician is often cited as one of the pioneers of “programmable” automata, of cybernetics, and of robotics.

– Galen (2nd Century AD) showed the purpose of human organs by analogy to machines built by man. In some way, he is also a pioneer of cybernetics.

–            ⋮

– 12th Century AD: android that opened the door and greeted when a bell was rung.

- ⋮
- 16th Century AD: mechanical lion (L. da Vinci) whose stomach would open and free lily flowers.
- ⋮

### 7.5.2. History

Emphasis is put on the reproduction of intellectual faculties or abilities that require different forms of learning.

- During the 17th Century, *mechanical philosophy* (or *mechanistic materialism*) tried to explain the world without ever referring to vital forces or vital causes. It believed that the method used to study the stars could be applied to physiology and human psychology.

- The circulatory system was discovered around 1628 (Harvey<sup>6</sup>), and this dented the vitalist theory (according to which vital phenomena are *irreducible* to physicochemical phenomena).

- 1641–1642: automatic adder (Pascal).

- 1673–1674: multiplication machine (Leibniz).

- During the 17th Century, Descartes viewed man as a machine.

- Thomas Hobbes (16th–17th Century) suggested that it could be considered that automata (meaning machines that move by themselves) have an *artificial life*.

- Leibniz had an eclectic mind and made connections between domains that seemed disconnected. He came up with the idea of a language for thought (*lingua philosophica* or *characteristica universalis*) in which everything that can be thought of could be transcribed, thus permitting reasoning to be (quasi)automated: the *calculus ratiocinator*<sup>7</sup>.

- 17th Century: probability calculus (probabilities are essential in many intelligent tasks: decision making on rational databases, etc.).

- 18th Century: the flute player, Vaucanson's duck (an engineer from Grenoble, France).

- 19th Century: Jacquard's weaving loom, punch cards.

- Ampère (1834) included *cybernetics* (science of direction) in "politics".

- 19th Century: development of probability calculus (Laplace, Poisson, etc.).

---

<sup>6</sup> And maybe even before M. Servet.

<sup>7</sup> Leibniz prophesied that once the calculus had been perfected, men of good faith who wanted to settle a question would take a pencil and paper and say: *Calculemus!*

– End of the 19th Century: Ramón Cajal discovered the nature of neurons and their interconnections.

– 1924: the word robot, from the Czech word *robota* meaning “forced labour” appears in a play by Karel Capek to denote artificial workers.

– 1927: Hilbert wrote: “The key idea of my proof theory is nothing more than the depiction of the activity of our intelligence, of listing and analyzing the rules that guide the way our thoughts really function”.

– Some physiologists (Belle, Young, Helmholtz, and others) convert to physics (H. Helmholtz believed that “no other forces than those of physics and chemistry are active within an organism”).

– ≈ 1936: Turing.

– ≈ 1936: Wiener *et al.* cybernetics (*science of direction and communication in living organisms and machines*).

– ≈ 1940: neural networks (MacCulloch, Pitts).

– Shortly after World War II, interdisciplinary exchanges (mathematicians, neurophysiologists, psychologists and sociologists) took place among other things on the possibility of imitating human intelligence.

– ≈ 1950: electronic turtles capable of recharging themselves.

– ≈ 1950: electronic mouse that could learn a path.

– . . . AI (the “official birth” of AI is supposed to have taken place in 1956).

– What is considered as the first AI program *Logic Theory Machine* (A. Newell, H. Simon), a program for theorem proving, is presented in 1956.

REMARK 7.6.– In the near future, recent advances in biology and in particular the study of the brain via medical imaging will probably influence the models that are used in AI, as well as its foundations and techniques. If that were to happen, it would be an item to add in the list above. □

## 7.6. Some undisputed themes in AI

– design of expert systems

– different kinds reasoning (inferences): deductive, inductive, abductive, probabilistic, non-monotonic, under uncertainty, etc.

– games (chess, go, etc.)

– knowledge representation

– learning

– robotics, vision, image analysis

– speech, writing recognition

– human–computer interaction

- natural language processing
- multi-agent systems
- planning
- constraint satisfaction
- computational linguistics
- neural networks

⋮

We have seen (section 7.2) that one of the approaches to studying AI was to try to mimic the way humans do things.

When we analyze (in particular by introspection) the way a human solves a problem, the most striking characteristics are the *diversity of approaches* that are used and the capability of humans to distinguish the “right” context and to use the “right” properties and relations for the problem under consideration.

This is particularly striking in so-called clever or elegant solutions.

In contrast, mechanical solutions would be qualified as “uniform”.

Furthermore, the etymology of the word “find” is worth mentioning here:

Find → twist, in Greek evolution, change, related to change,

which seems to suggest that to find a solution, it is necessary to consider a problem by “twisting” it, i.e. by analyzing it from different angles.





## Chapter 8

# Inference

Inference is one of the most important intelligent activities of a human being. We will study different forms of inference and several aspects of its formalization. According to the dictionary:

Inference: every operation leading to the acceptance of a proposition whose truth is not directly known, thanks to its relationship to other propositions known to be true. This relationship can be such that the inferred proposition is judged to be necessary, or only plausible.

*Inference* is therefore the most general term of which *reasoning*, *deduction*, *induction*, etc. are instances.

Etymologically: to offer  $\rightarrow$  to infer: to carry into - to put forward

Inference is an underlying activity of almost all our other activities. Something as common as vision, for example, uses it (probably without the knowledge of doing so). When we see someone we know, but who has let his beard grow, is wearing glasses (and did not use to before), has put on some weight or lost some weight, etc. we still recognize him, although from the point of view of appearance, this is someone we have never seen.

There are many different forms of inference, among which are inductive (i.e. general conclusion from particular facts), by analogy, from testimonials, from memories, probable inference, statistical inference, non-monotonic, etc.

We begin with the study of one of the most common forms of inference (often carried out unconsciously), which is often mentioned in this book.

### 8.1. Deductive inference

As usual, a first approach can rely on the definition in the dictionary.

Deduction: operation by which, starting from one or several propositions taken as premises, we rigorously conclude a proposition that is a necessary consequence, in accordance with logical rules.

Etymologically: to conduct  $\rightarrow$  to lower  $\rightarrow$  to deduce

We shall give several classical types of deductions, focusing on the key problems that arise in the process of their automation, through examples that shed light on these problems.

The typical example of a deduction is the syllogism, which we have mentioned in section 2.2.

The following example is a less known deduction type that leads to interesting problems.

#### The sorites

Argument consisting of a mass of premises. Polysyllogism, where each conclusion is used as a premise of the following syllogism. In more modern terms, we would say that this is a reasoning with a *linear* strategy (see example 5.28).

EXAMPLE 8.1.–  $A$  is  $B$ ,  $B$  is  $C$ ,  $C$  is  $D$ ,  $D$  is  $E$ ; therefore  $A$  is  $E$ . □

The name *sorites* is especially used in the following very old reasoning (see also section 10.2) given by due to Eubulides (see digression 2.2).

Does a heap of grains remain a heap when one grain is removed?

Consider a heap of grains: if we remove one grain, it remains a heap, if we remove another grain, it remains a heap...

Conclusion: a unique grain is a heap... but [everyone agrees that] it actually is not!

It is sometimes presented as follows:

A grain is not a heap, neither are two grains, ... After how many grains do we get a heap?

An aspect that is often neglected is that of *partial conclusions*. It should be clear that a given strategy, when applied to a problem, can lead to the same conclusion going through different partial conclusions, *depending on the order in which the premises are considered*. The following example given by L. Carroll illustrates this.

EXAMPLE 8.2.– (partial conclusions). Deduce the conclusion with a linear strategy (see example 5.28), considering different orders on the premises (partial conclusions are written in bold).

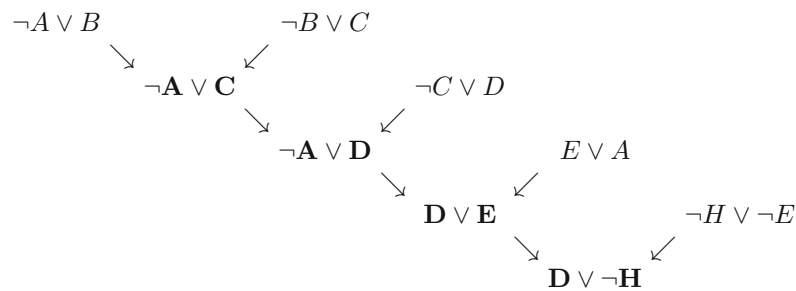
- No  $A$  is not  $B$ .
- Every  $B$  is  $C$ .
- Every  $C$  is  $D$ .
- No not  $E$  is not  $A$ .
- Every  $H$  is not  $E$ .
- Therefore:
- Every  $H$  is  $D$ .

We translate into PL (sometimes the categorical propositions of syllogisms are formalized in this logic) and into clausal form. For deductions, we use the resolution rule with a linear strategy and two different orders.

$A \Rightarrow B$	1	$\neg A \vee B$
$B \Rightarrow C$	2	$\neg B \vee C$
$C \Rightarrow D$	3	$\neg C \vee D$
$\neg E \Rightarrow A$	4	$E \vee A$
$H \Rightarrow \neg E$	5	$\neg H \vee \neg E$
$H \Rightarrow D$	6	$\neg H \vee D$

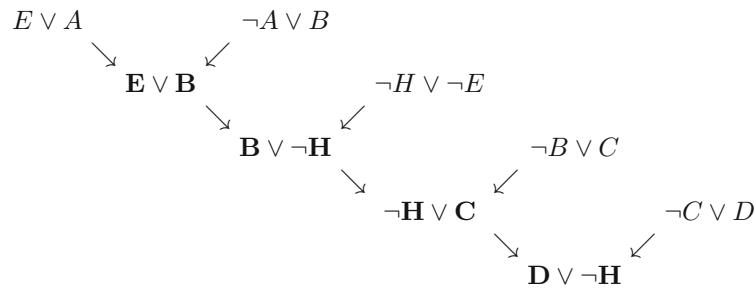
Note that the conclusion (6) is not negated, we do not try to derive  $\square$  but (6)<sup>1</sup>.

With the order 1,2,3,4,5:



<sup>1</sup> The resolution method is correct (see corollary 3.1), but we have only proved its *refutational* completeness (see exercise 3.34).

With the order 4,1,5,2,3:



□

Sometimes, as L. Carroll wondered, it is interesting to obtain all the logical consequences of the premises. The following example shows that this can be really useful<sup>2</sup>.

EXAMPLE 8.3.– (beware of hidden consequences!). From the following premises:

- 1) Every individual suitable to be a member of Parliament who does not spend all his time making speeches is a benefactor of the people.
- 2) People with a clear mind and who express themselves well have received a decent education.
- 3) A woman worthy of appraisal is a woman who knows how to keep a secret.
- 4) Those who do favors for the people but do not use their influence for praiseworthy purposes are not suitable to be members of Parliament.
- 5) Those who are worth their weight of gold and are not worthy of appraisal are always unpretentious.
- 6) Benefactors of the people who use their influence for praiseworthy purposes are worthy of appraisal.
- 7) Those who are unpopular and are not worth their weight in gold do not know how to keep a secret.
- 8) Those who know how to talk for hours and hours and are not suitable to be members of Parliament are worthy of appraisal.
- 9) Every individual who knows how to keep a secret and is unpretentious is a benefactor of the people whose memory will last forever.
- 10) A woman who does favors for the people is always popular.

<sup>2</sup> We can also imagine that we want to obtain all the consequences of a set of laws.

11) Those who are worth their weight of gold, who do not stop making speeches and whose memory lasts forever are precisely those whose photography can be seen in all shop windows.

12) A woman who does not have a clear mind and has not received a proper education is unsuitable to become a member of Parliament.

13) Every individual who knows how to keep a secret and does not know how to always make speeches can be sure to be unpopular.

14) An individual with a clear mind, who has influence and uses it for praiseworthy purposes is a benefactor of the people.

15) A benefactor of the people who is unpretentious is not the kind of person whose photography is shown in all shop windows.

16) Those who know how to keep a secret and use their influence for praiseworthy purposes are worth their weight in gold.

17) Someone who does not know how to express himself and is incapable of influencing others cannot be a woman.

18) Those who are popular and worthy of appraisal are either benefactors of the people or unpretentious.

We can deduce, among other things:

A woman is not suitable to be a member of Parliament! □

REMARK 8.1.– In PL, starting with a finite set of premises, we *always* obtain by resolution a finite set of conclusions. This is not the case in FOL: it suffices to consider the set of conclusions of the premises in the following example. □

EXAMPLE 8.4.– (complexity of proofs). Consider the following reasoning:

$$\frac{\begin{array}{c} P(a) \\ \forall x(P(x) \Rightarrow P(f(x))) \end{array}}{P(f^{2^k}(a))} \quad k \in \mathbb{N}$$

We analyze the complexity of two proofs by resolutions that are obtained with different strategies for  $k = 3$ :

- 1)  $P(a)$
- 2)  $\neg P(x) \vee P(f(x))$
- 3)  $\neg P(f^8(a))$
- 4)  $P(f(a))$  (1, 1) – (2, 1)  $\{x_2 \leftarrow a\}$
- 5)  $P(f^2(a))$  (4, 1) – (2, 1)  $\{x_3 \leftarrow f(a)\}$
- 6)  $P(f^3(a))$  (5, 1) – (2, 1)  $\{x_4 \leftarrow f^2(a)\}$

- 7)  $P(f^4(a))$              $(6, 1) - (2, 1) \{x_5 \leftarrow f^3(a)\}$
- 8)  $P(f^5(a))$              $(7, 1) - (2, 1) \{x_6 \leftarrow f^4(a)\}$
- 9)  $P(f^6(a))$              $(8, 1) - (2, 1) \{x_7 \leftarrow f^5(a)\}$
- 10)  $P(f^7(a))$             $(9, 1) - (2, 1) \{x_8 \leftarrow f^6(a)\}$
- 11)  $P(f^8(a))$             $(10, 1) - (2, 1) \{x_9 \leftarrow f^7(a)\}$
- 12)  $\square$                      $(11, 1) - (3, 1)$

The number of steps in the proof (independently of the choice between forward and backward chaining) is:

$$\boxed{2^k + 1}$$

A less costly proof:

- 1)  $P(a)$
- 2)  $\neg P(x) \vee P(f(x))$
- 3)  $\neg P(f^8(a))$
- 4)  $\neg P(x) \vee P(f^2(x))$      $(2, 2) - (2, 1) \{x_2 \leftarrow f(x_3)\}$
- 5)  $\neg P(x) \vee P(f^4(x))$      $(4, 2) - (4, 1) \{x_4 \leftarrow f^2(x_5)\}$
- 6)  $\neg P(x) \vee P(f^8(x))$      $(5, 2) - (5, 1) \{x_6 \leftarrow f^4(x_7)\}$
- 7)  $P(f^8(a))$                  $(6, 1) - (1, 1) \{x_8 \leftarrow a\}$
- 8)  $\square$                          $(7, 1) - (3, 1)$

The number of steps in the proof is:

$$\boxed{k + 2}$$

... the difference is essential  $\square$

EXAMPLE 8.5.– (appearances are deceptive). Consider the following reasoning:

$$\frac{\begin{array}{l} \forall x \forall y \forall z (P(x, y) \wedge P(y, z) \Rightarrow P(x, z)) \\ \forall x \forall y \forall z (Q(x, y) \wedge Q(y, z) \Rightarrow Q(x, z)) \\ \forall x \forall y (Q(x, y) \Rightarrow Q(y, x)) \\ \forall x \forall y (P(x, y) \vee Q(x, y)) \end{array}}{\forall x \forall y (P(x, y) \vee \forall x \forall y Q(x, y))}$$

In other words, given a transitive relation  $P$  and a symmetric and transitive relation  $Q$ , and given that any two elements are related either by  $P$  or by  $Q$ , prove that  $P$  is total or  $Q$  is total.

We decide to prove that this reasoning (or theorem) is correct using the resolution rule. We thus translate into clausal form (after negating the conclusion and Skolemization, we obtain clauses 5 and 6).

- 1)  $\neg P(x, y) \vee \neg P(y, z) \vee P(x, z)$
- 2)  $\neg Q(x, y) \vee \neg Q(y, z) \vee Q(x, z)$
- 3)  $\neg Q(x, y) \vee Q(y, x)$
- 4)  $P(x, y) \vee Q(x, y)$
- 5)  $\neg P(a, b)$
- 6)  $\neg Q(c, d)$

We must find a refutation by resolution of the set of clauses above.

Note that this set of clauses belongs to a *decidable fragment of FOL* (finite Herbrand universe), so we are sure that the search halts and that the set is either unsatisfiable or satisfiable.

The proposed solution involves a typically human “trick”:

– we know that unit clauses are useful (the resolvent contains less literals than the non-unit parent clause);

– we therefore try to generate unit clauses (we will also use *hyperresolution* to obtain clauses 10, 11, 12, 16, 12', 14');

– for any closed term  $\bar{t}$  in the Herbrand universe of a set of clauses  $S$  and for any predicate  $P$  occurring in  $S$ , we have either  $P(\bar{t})$  or  $\neg P(\bar{t})$ .

- 7)  $Q(a, b)$  (4, 1) – (5, 1)  $\{x_4 \leftarrow a, y_4 \leftarrow b\}$
- 8)  $Q(b, a)$  (3, 1) – (7, 1)  $\{x_3 \leftarrow a, y_3 \leftarrow b\}$
- 9)  $Q(c, a)$  ASSUMED
- 10)  $Q(c, b)$  (2, 1) – (9, 1) – (7, 1)  $\{x_2 \leftarrow c, y_2 \leftarrow a, z_2 \leftarrow b\}$
- 11)  $\neg Q(a, d)$  (2, 1) – (6, 1) – (9, 1)  $\{x_8 \leftarrow c, y_8 \leftarrow a, z_8 \leftarrow d\}$
- 12)  $\neg Q(b, d)$  (2, 1) – (6, 1) – (10, 1)  $\{x_{14} \leftarrow c, y_{14} \leftarrow b, z_{14} \leftarrow d\}$
- 13)  $\neg Q(d, b)$  (3, 2) – (12, 1)  $\{x_9 \leftarrow d, y_9 \leftarrow b\}$
- 14)  $P(d, b)$  (4, 2) – (13, 1)  $\{x_4 \leftarrow d, y_4 \leftarrow b\}$
- 15)  $P(a, d)$  (4, 2) – (11, 1)  $\{x_{10} \leftarrow a, y_{10} \leftarrow b\}$
- 16)  $P(a, b)$  (1, 1) – (15, 1) – (1, 2) – (14, 1)  $\{x_1 \leftarrow a, y_1 \leftarrow d, z_1 \leftarrow b\}$
- 17)  $\square$  (5, 1) – (16, 1)
- 9')  $\neg Q(c, a)$  ASSUMED
- 10')  $\neg Q(a, c)$  (3, 2) – (9', 1)  $\{x_3 \leftarrow a, y_3 \leftarrow c\}$
- 11')  $P(a, c)$  (4, 2) – (10', 1)  $\{x_4 \leftarrow a, y_4 \leftarrow c\}$
- 12')  $\neg P(c, b)$  (1, 3) – (5, 1) – (1, 1) – (11', 1)  $\{x_1 \leftarrow a, y_1 \leftarrow c, z_1 \leftarrow b\}$

$$\begin{array}{ll}
13') Q(c, b) & (4, 1) - (12', 1) \{x_{10} \leftarrow c, y_{10} \leftarrow b\} \\
14') Q(c, a) & (2, 1) - (13', 1) - (2, 2) - (8, 1) \{x_2 \leftarrow c, y_2 \leftarrow b, z_2 \leftarrow a\} \\
15') \square & (9', 1) - (14', 1). \quad \square
\end{array}$$

REMARK 8.2.– The trick that was used corresponds to the law of excluded middle in some so-called *natural deduction* systems:

$$\begin{array}{c}
[A] \quad [\neg A] \\
\\
\frac{B \quad B}{B}
\end{array}$$

The formulas between [ ] are *discharged* or *cancelled*: they are no longer useful (see digression 3.5).

The rule reads: if from  $A$  we can deduce  $B$  and from  $\neg A$  we can deduce  $B$  then  $B$ .  $\square$

## 8.2. An important concept: clause subsumption

In informal reasonings, to derive a conclusion (or verify that a reasoning is correct), we frequently use the fact that a property or relation satisfied by all the objects of a universe is also satisfied by particular objects of this universe (see, e.g. example 5.1). We also use the fact that when we have a disjunction, whatever makes a subset of the disjuncts true also makes the entire disjunction true.

These two very simple remarks are used in definition 8.1. According to the dictionary:

To subsume: fact of considering a thing as being a member of a whole. Considering an individual as being a member of a species, or a species as being a member of a genus; considering a fact as the application of a law.

*Classification*,<sup>3</sup> which deals with the domain of *systematics* (or *taxonomy*) and is used especially in botany and zoology, beginning with the work of the great Swedish botanist Carl von Linné (18th Century), structures the objects considered in these sciences with a partial order relation (corresponding to inclusion): race, species, genus, family, order, class, type and reign.

---

<sup>3</sup> Which was considered from the beginning of modern science (for example, by Francis Bacon (1561–1626)) as being a part of the scientific method.



There exist many knowledge representation languages (among which the classical languages are KL-ONE, KRYPTON, LILOG, etc.). Languages such as KL-ONE, which was frequently used in natural language-processing systems, enable us to describe concepts using unary predicates.

Knowledge is separated into:

- a terminological part (concept definition): the T-box;
- an assertional part (database): the A-box.

The language that is used by the T-box corresponds to a fragment of FOL.

Subsumption determines the order relation (with  $\subseteq$  as an order relation) between concepts.

Concept  $C$  is subsumed by concept  $D$  if all the instances of  $C$  are necessarily instances of  $D$ , meaning that the extension of  $C$  is interpreted as a subset of the extension of  $D$  (see digression 8.1).

This very short motivation is meant to show that to dispose of a method (if possible a *decision procedure*) to decide whether relations hold between certain formulas can be very useful in knowledge manipulation.

DEFINITION 8.1.– (clause subsumption). A clause  $C$  ( $\theta$ )-subsumes or simply subsumes a clause  $D$  iff:

- i) there exists a substitution  $\theta$  such that:

$\theta C$  is a sub-clause of  $D$  (if clauses are considered as disjunctions of literals).

or:

$\theta C \subseteq D$  (if clauses are considered as sets of literals).

We will write  $C \leq_s D$ .

In general, we impose that:

- ii)  $\text{number-of-literals}(C) \leq \text{number-of-literals}(D)$

(or  $\text{card}(C) \leq \text{card}(D)$ )

We say that  $C$  is more general than  $D$ . This terminology is easy to understand if we take into account the fact that  $C$  contains more (universally quantified) variables than  $D$  (see also exercise 8.1 a)).

REMARK 8.3.– This definition applies to PL with  $\theta = \emptyset$  (see rule R-4 of the Davis–Putnam method, section 3.5).

Condition (ii) is sometimes justified by the fact that:

$$P(x, a) \vee P(b, y) \text{ subsumes } P(b, a)$$

$$(\theta = \{x \leftarrow b, y \leftarrow a\})$$

This definition can be considered as somewhat “unnatural”, because we may keep a more complex object in the inference process.  $\square$

EXAMPLE 8.6.–  $P(x) \vee Q(y)$  subsumes  $P(a) \vee Q(b) \vee R(u, z)$

$$\theta = \{x \leftarrow a, y \leftarrow b\}.$$

$\square$

EXERCISE 8.1.–

a) Prove that if  $C$  subsumes  $D$ , then  $C \models D$ .

b) Is the problem *does  $C$  subsume  $D$*  decidable?  $\square$

### 8.2.1. An important problem

Can we say: if  $C \models D$ , then  $C$  subsumes  $D$ ?

Consider:

$$C: \neg P(x, y) \vee \neg P(y, z) \vee P(x, z)$$

$$D: \neg P(a, b) \vee \neg P(b, c) \vee \neg P(c, d) \vee P(a, d)$$

We prove that  $C \models D$  using the resolution method; we must thus show that  $C \cup \{\neg D\} \vdash_{\mathcal{R}} \square$

$$\neg D: P(a, b) \wedge P(b, c) \wedge P(c, d) \wedge \neg P(a, d)$$

(i.e. there are four clauses, see also exercise 5.3 (1))

$$1) \neg P(x, y) \vee \neg P(y, z) \vee P(x, z)$$

$$2) P(a, b)$$

$$3) P(b, c)$$

$$4) P(c, d)$$

$$5) \neg P(a, d)$$

- |   |  |
|---|--|
| 6) $\neg P(a, y) \vee \neg P(y, d)$     | (5,1)-(1,3) $\{x \leftarrow a, z \leftarrow d\}$     |
| 7) $\neg P(a, c)$                       | (6,2)-(4,1) $\{y \leftarrow c\}$                     |
| 8) $\neg P(a, y_1) \vee \neg P(y_1, c)$ | (7,1)-(1,3) $\{x_1 \leftarrow a, z_1 \leftarrow c\}$ |
| 9) $\neg P(a, b)$                       | (8,2)-(3,1) $\{y_1 \leftarrow b\}$                   |
| 10) $\square$                           | (9,1)-(2,1)  |

... but  $C$  does not ( $\theta$ -) subsume  $D$ :

indeed, the desired substitution  $\theta$  must contain  $\{x \leftarrow a, z \leftarrow d\}$ , which requires either  $y \leftarrow b$ ;  $\theta C$  will then contain  $\neg P(b, d)$ , and the subsumption is impossible; or  $y \leftarrow c$ ;  $\theta C$  will then contain  $\neg P(a, c)$ , and the subsumption is impossible.

We give another example showing that if  $C$  subsumes  $D$ , then  $D$  is a logical consequence of  $C$ , but that the converse does not hold.

$$C: \neg P(f(x)) \vee P(x)$$

$$D: \neg P(f(f(a))) \vee P(a)$$

We use the resolution method to show that  $C \models D$

$\{C, \neg D\}$  is the set of clauses 1, 2, 3 below:

- 1)  $\neg P(f(x)) \vee P(x)$
- 2)  $P(f(f(a)))$
- 3)  $\neg P(a)$
- 4)  $\neg P(f(a))$       (1,2)-(3,1)  $\{x \leftarrow a\}$
- 5)  $\neg P(f(f(a)))$       (1,2)-(4,1)  $\{x_1 \leftarrow f(a)\}$
- 6)  $\square$       (2,1)-(5,1)

REMARK 8.4.– Given two clauses  $C$  and  $D$ , the problem  $C \models^? D$  is undecidable.

This answer, together with the answer to exercise 8.1 (a), also allow us to show that clause subsumption and logical consequences between clauses are not equivalent.  $\square$

The following theorem, which is admitted without a proof, indirectly gives the key (auto-resolvent clauses) to the non-equivalence between subsumption and logical consequences between clauses.

**THEOREM 8.1.**– *If  $C$  is a clause that is not auto-resolvent and  $D$  is a non-tautological clause, then  $C \models D$  iff  $C$  subsumes  $D$ .*

**DEFINITION 8.2.**– *Let  $C$  be a clause. We denote by:*

$C^+$  *the set of positive literals in  $C$ ;*

$C^-$  *the set of negative literals in  $C$ .*

$C$  *is ambivalent iff there exists a predicate symbol in  $C$  that occurs in  $C^+$  and in  $C^-$ .*<sup>4</sup>

An immediate consequence of this definition is that a positive (respectively, negative) clause cannot be ambivalent.

**THEOREM 8.2.**– *Let  $C$  and  $D$  denote two clauses. If  $D$  is non-tautological and  $C \models D$ , then  $C^+$  subsumes  $D^+$  and  $C^-$  subsumes  $D^-$ .*

**PROOF.**– As  $C^+$  is a sub-clause of  $C$ , every model of  $C^+$  is a model of  $C$  (because clauses are *disjunctions* of literals). The same reasoning shows that every model of  $C^-$  is a model of  $C$ .

$C^+$  (respectively,  $C^-$ ) is not auto-resolvent, as it only contains positive (respectively, negative) literals.

By application of theorem 8.1:

$C^+ \models D$  iff  $C^+$  subsumes  $D$

$C^- \models D$  iff  $C^-$  subsumes  $D$ .

But  $C^+$  (respectively,  $C^-$ ) can only be a sub-clause of  $D^+$  (respectively,  $D^-$ ); hence,

$C^+$  subsumes  $D^+$  and

$C^-$  subsumes  $D^-$ . □

**THEOREM 8.3.**– *Consider two clauses  $C$  and  $D$ . If  $D$  is not ambivalent, then  $C \models D$  iff  $C$  subsumes  $D$ .*

---

<sup>4</sup> Note that an auto-resolvent clause is necessarily ambivalent, but that an ambivalent clause is not necessarily auto-resolvent, for example,  $P(a) \vee \neg P(b)$ .

PROOF.— We assume that  $C \models D$ .

If  $D$  is not ambivalent, then it cannot be a tautology (because all its literals can be evaluated to **F**). Thus (theorem 8.2):

(\*)  $C^+$  subsumes  $D^+$  and  $C^-$  subsumes  $D^-$ .

$C$  cannot be auto-resolvent: that would require a predicate symbol, say  $P$ , to occur in  $C^+$  and  $C^-$ , but then (definition of subsumption and (\*)),  $P$  should occur in  $D^+$  and  $D^-$ , which is impossible as  $D$  is not ambivalent.

Therefore, by applying theorem 8.1, we obtain:

$C \models D$  iff  $C$  subsumes  $D$ . □

REMARK 8.5.— For the same Prolog program, if we have two questions  $C$  and  $D$ , as they are both negative, theorem 8.3 applies; hence, we have a decidable test to know whether  $C \models D$ , meaning that it suffices to have answered to  $C$  to answer to  $D$ . □

Theorems 8.1, 8.2, and 8.3 enable us to obtain the procedure  $D$  LOG-CONS  $C?$  to test whether a clause  $D$  is the logical consequence of a clause  $C$ . As we mentioned in remark 8.4, this is an undecidable problem, so the procedure may not terminate (because to procedure “Consequence” that is used in procedure “D log-cons C?”).

```

procedure D LOG-CONS C ?
input: two clauses
output: (in case of termination) true or false
begin
if tautology (D)  $\vee$  subsumes (C, D)
then return true
else % (theorem 8.1 + theorem 8.3)
    if  $\neg$  (ambivalent(D)  $\wedge$  auto-resolvent (C))
    then return false
    else % theorem 8.2
        if  $\neg$  (subsumes (C+, D+)  $\wedge$  subsumes (C-, D-))
        then return false
        else return Consequence (C,D)
end

```

Figure 8.1. Procedure for testing logical consequence between clauses

EXAMPLE 8.7.— As mentioned previously, we will say that clause  $C$  subsumes clause  $D$  instead of: clause  $C \theta$  subsumes clause  $D$ .

- 1)  $C_1 \leq_s C$  and  $C_2 \leq_s C : C_1 \vee C_2 \leq_s^? C$
- 2)  $C^- \leq_s C_1^-$  and  $C^+ \leq_s C_1^+ : C \leq_s^? C_1$
- 3) is relation  $\leq_s$  a quasi-order, i.e. a reflexive and transitive relation?
  - 1) No, take  $C_1 : P(x); C_2 : P(f(x))$  and  $C : P(a) \vee P(f(f(a)))$ .
  - 2) No, take  $C : \neg P(x) \vee P(f(x))$  and  $C_1 : \neg P(a) \vee P(f(f(a)))$ .
  - 3) Yes.

reflexive: take  $\sigma$ : the identity:  $\sigma C \subseteq C$

transitive:

$$C_1 \leq_s C_2 \text{ and } C_2 \leq_s C_3$$

there exist  $\sigma_1, \sigma_2$  such that  $\sigma_1 C_1 \subseteq C_2$  and  $\sigma_2 C_2 \subseteq C_3$

therefore:

$$\sigma_2(\sigma_1 C_1) \subseteq C_3$$

hence, there exists  $\theta = \sigma_2 \circ \sigma_1 = \sigma_1 \sigma_2$

such that  $\theta C_1 \subseteq C_3$ ; i.e.:

$$C_1 \leq_s C_3. \quad \square$$

DIGRESSION 8.1.– (ontologies and DL). *Description logics* are languages for ontologies. As we mentioned in section 1.2, an ontology can be defined as *an explicit specification of a conceptualization*. In other words, an ontology is a(n abstract) model of a part of reality (the world) that we are interested in. This part of reality is represented by pertinent concepts (properties) and by relations between these concepts. This in a *formalizable* language<sup>5</sup>.

Ontologies have naturally taken a very important place in computer science for different kinds of information-processing systems, and a privileged field of application is the semantic web.

In order for them to be useful, ontology languages must have some *inference* capabilities, in particular, to detect non-contradiction between concepts or their relations.

---

<sup>5</sup> Note the similarity with, e.g. computer modeling, databases, object-oriented programming, etc.

Why the name DL? The concepts that are relevant to the domain under consideration are specified by *descriptions* of atomic concepts using properties (unary predicates) and relations (*atomic roles*) (binary predicates). *Logics* because their semantic is defined in a similar way to that used in logic.

*Constructors* are the logical constants (see definition 5.1) and some types of quantifiers.

DLs generally offer the possibility to name (complex) descriptions, constraints about the inclusion of concepts, and assertions of properties and relations between particular objects (note the analogy with logic programming).

The *subsumption* algorithm permits us to detect the inclusion between concepts. The definition of subsumption corresponds to the one in definition 8.1: concept  $D$  is subsumed by concept  $C$  iff every instance (i.e. particular case) of  $D$  is an instance of  $C$  (for example, the man concept is subsumed by the animal concept).

DLs can also detect the (in)consistency (i.e. the contradiction (or non-contradiction)) of the set of assertions and definitions. The consistency (or inconsistency) of a set of assertions can be proved by exhibiting a model (or the impossibility of constructing one), the method of semantic tableaux is a tool that is naturally used to automate such a process.

DLs are closely related to modal logics (see section 10.3).

The syntax and semantics of DLs are defined using methods similar to those used for FOL, which is normal as most DLs correspond to decidable fragments of FOL.

As for any automated inference problem, the trade-off between expressive power and decidability (AND complexity of the decision algorithm) must be appropriately resolved for DLs.  $\square$

### 8.3. Abduction

Aristotle viewed abduction as a specific kind of reasoning<sup>6</sup>. Much more recently, C.S. Peirce (19th Century, beginning of the 20th Century) identified abduction as a specific form of reasoning based on principles different from the standard principles such as deduction and induction (see section 8.4). He used *hypothesis*, *hypothesis inference*, *abduction*, and *retroduction* as synonyms.

---

<sup>6</sup> Aristotle named *abduction* a syllogism in which the major was certain and the minor was only plausible.

He acknowledged the utility of abduction, but also the difficulty of its theoretical justification:

Now nothing justifies a retroductive inference excepts its affording an explanation of the facts.

Sometimes it is formalized with the following rule (which is incorrect from the point of view of deductive logic):

$$Abd : \frac{\beta \quad \alpha \Rightarrow \beta}{\alpha}$$

$\alpha$  is an *explanation* of  $\beta$ .

The intuitive reason why this rule is incorrect is clear. For example, we know that flu causes fever, but when we have a fever, it can be due to another illness.

Peirce clearly showed the importance of abduction in scientific reasoning (giving as a paradigm the discovery of Kepler's laws).

We find fish fossils on dry land. We then suppose that, *in the past, the sea used to cover the land*. This explanation (obviously not certain and not unique) is generally added to a *corpus* of knowledge that attributes some weight to the explanation.

Abduction is analogous to what are called, in particular in mathematics, *inverse problems*; these are *ill-posed problems*: theorems are formalized and we wonder what axioms are necessary to prove these theorems (or we look for the parameters that make a law true).

### 8.3.1. *Discovery of explanatory theories*

The context: we have a theory or a repository of knowledge at our disposal (set of non-contradictory formulas, in particular, an empty theory).

We observe: particular events. Some objects have some properties or are related to other objects, these are the positive examples; other objects do not have some properties or they are not related to some other objects, these are the negative examples.

The problem: generate hypotheses that enable us to explain the observations, i.e. formulas from which the observations can be deduced.

This description corresponds exactly to the scientific work in natural sciences.



The justification of the method for the discovery of explanations raises very profound philosophical problems. There are considerable differences of opinion on the essential characteristics of scientific explanation.

Notation:

$C$  (also noted  $K$ ): theory or knowledge on the domain of the observed facts.

$E^+$ : set of positive examples ( $e_i^+$ );

$E^-$ : set of negative examples ( $e_i^-$ );

$H$ : set of generated hypotheses.

### 8.3.1.1. Required conditions

We give the three following formulations **A**, **B**, and **C**, which represent the conditions that are required of explanatory theories (as for the resolution method,  $\square$  means contradiction).

**A:**

$$1) K \cup E^- \not\models \square$$

% Satisfiability *a priori*

$$2) K \not\models E^+$$

% Necessity *a priori*

$$3) K \cup H \models E^+$$

% Completeness with respect to  $E^+$

$$4) K \cup H \cup E^- \not\models \square$$

% Consistency with respect to  $E^-$

**B (with  $E = E^+$ ):**

$$1) K \cup H \models E$$

% The hypothesis explains the examples

$$2) K \cup E \not\models \neg H$$

% The hypothesis is not in contradiction with the theory

3)  $K \not\models H$

% The hypothesis is not redundant

4)  $K \not\models E$

% The theory does not explain  $E$  (necessity of  $H$ )

**C:**

1)  $K \not\models E^+$ , i.e.  $C \not\models e_j^+$  for (at least one)  $e_j^+ \in E^+$

% The observed facts cannot be explained by the theory alone

2)  $K \not\models H$

%  $H$  is not redundant

3)  $K \cup H \not\models e_i^-$  for all  $e_i^- \in E^-$

% The negative examples (counter examples) cannot be explained by the addition of  $H$

4)  $K \cup H \models e_j^+$  for all  $e_j^+ \in E^+$

% The addition of  $H$  enables us to explain the observed facts

Condition (4), which is sometimes formalized by:

$$K \cup H \cup E^+ \not\models \square$$

corresponds to what is known in philosophy of science as the *coherence theory of truth*, of which one of the versions is: *the truth of a (true) proposition consists in its consistency with a given set of propositions.*

REMARK 8.6.— The relation of non-logical consequence ( $\not\models$ ) is used. Its treatment is not simple in FOL, where the set of non-logical consequences of a set of formulas is not recursively enumerable (if it were, FOL would be decidable).  $\square$

The following example shows some peculiarities of abduction (meaning that the proposed explanations generally depend on the deduction strategy that is used).

EXAMPLE 8.8.– (abduction and strategies). We wonder whether clause 6 given below is a logical consequence of clauses 1–5. If this is not the case, what premises can be added so that 6 becomes a logical consequence of the new set of premises?

$$\begin{array}{l}
 1) \quad \neg Q \vee \neg R \vee P \\
 2) \quad \neg S \vee Q \\
 3) \quad \neg S \vee \neg U \vee R \\
 4) \quad \neg V \vee \neg W \vee R \\
 5) \quad S \\
 \hline
 6) \quad P
 \end{array}$$

It is trivial to verify that 6 is not a logical consequence of 1–5 (counter example:  $\{S, Q\}$ , i.e.  $S$  and  $Q$  are interpreted to **T** and all other propositional symbols are interpreted to **F**).

To discover the desired premises, we apply resolution with two different strategies, which will lead to two different explanations.

$$\begin{array}{l}
 7) \quad \neg R \vee \neg S \vee P \quad (1, 1) - (2, 2) \\
 8) \quad \neg R \vee P \quad (7, 2) - (5, 1)
 \end{array}$$

Explanation-1:  $R$ .

(We cannot deduce  $R$  because that would require to eliminate either  $\neg U$  or  $\neg V$  and  $\neg W$ , and all three of them are pure literals).

$$\begin{array}{l}
 7') \quad \neg U \vee R \quad (3, 1) - (5, 1) \\
 8') \quad \neg Q \vee \neg U \vee P \quad (7', 2) - (1, 2) \\
 9') \quad \neg S \vee \neg U \vee P \quad (8', 1) - (2, 2) \\
 10') \quad \neg U \vee P \quad (9', 1) - (5, 1)
 \end{array}$$

Explanation-2:  $U$ . □

EXAMPLE 8.9.– (conditional answers). In logic programming, abduction could be used to give conditional answers when the program does not provide an answer.

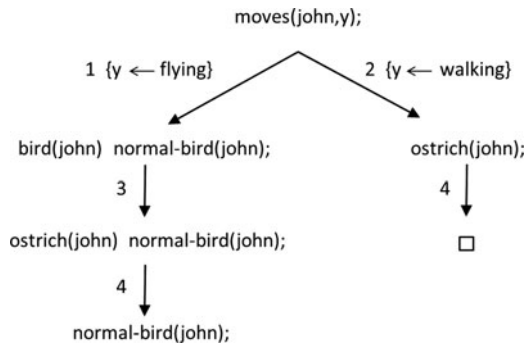
Consider a program (often used as an example) that describes the characteristics of some animals.

- 1)  $\text{moves}(x, \text{flying}) \rightarrow \text{bird}(x) \text{ normal-bird}(x);$
- 2)  $\text{moves}(x, \text{walking}) \rightarrow \text{ostrich}(x);$
- 3)  $\text{bird}(x) \rightarrow \text{ostrich}(x);$
- 4)  $\text{ostrich}(\text{john}) \rightarrow;$

and we ask the question

`moves(john,y);`

We give the execution tree:



The unconditional answer (i.e. the standard answer) is thus:

`moves(john,walking)`

The conditional answer would be:

**if** `normal-bird(john)` **then** `moves(john,flying)`. □

#### 8.4. Inductive inference

It seems like no one found a trace of an induction problem before the second half of the 17th Century. Induction seems to have been a problem forgotten in philosophy.

Now, the induction problem is unanimously recognized as an essential problem in knowledge theory.

There is currently a wide consensus to the fact that the induction problem was posed by the philosopher David Hume<sup>7</sup>.

Hume realizes that the information we have (that we receive, that impresses us) of the world is made of little fragments about the present or the past, and this is the only foundation that supports our general knowledge.

In other words, our *data* are made of *particular facts*.

Induction is a *conjectural* inference.

<sup>7</sup> It should be noted that *induction* is already mentioned in the treatise “*Logic: or, The Art of Thinking*”, published in 1662 during the emerging of probability theory: “We name induction, when the search for many particular things leads us to the knowledge of a general truth”.

Etymologically: to lead  $\rightarrow$  inducere  $\rightarrow$  to let in  $\rightarrow$  to bring

The first questions that come to mind:

- how can we, starting from particular experiences, establish laws that go beyond experience?
- can inductive inference be justified rationally?

These questions are very deep and habits often prevent us from analyzing them. If we are asked whether the sun will rise tomorrow, we will probably answer: “of course, as it has always been that way”.

But it is well known that it is delicate to make such generalizations, which are sometimes correct and sometimes incorrect. The following example (given by L. Euler (1707–1783)) shows what an *incorrect* inductive inference can be.

EXAMPLE 8.10.– (*incorrect* inference). We want to verify that, for  $0 \leq n \leq 39$ , the polynomial  $n^2 + n + 41$  yields the sequence of prime numbers 41, 43, 47, 53, 61, . . . We may believe that we have found a formula allowing us to generate all prime numbers greater than 41. But this is only true for  $0 \leq n \leq 39$ . For  $n = 40$  we get:  $40^2 + 40 + 41 = 41^2$ , which is obviously not a prime number.  $\square$

Nevertheless, induction has an undeniable heuristic value, in mathematics, for example L. Euler stated that most properties on numbers had been discovered starting from the observation of examples and induction.

As the great mathematician Henri Poincaré said:

As in other sciences, mathematics can therefore proceed from the particular to the general.

By noticing the analogies between recursion and induction, Poincaré said:

Induction applied to physics is always uncertain, because it relies on the belief in a general order of the Universe, order that is outside of us. On the contrary, mathematical induction, i.e. proofs by recursion, imposes itself necessarily, because it is only the assertion of a property of the mind itself.

It is necessary to point out an essential difference with deductive inference.

#### 8.4.1. *Deductive inference*

$$\frac{\begin{array}{l} \text{Every } A \text{ is a } B \\ a \text{ is an } A \end{array}}{a \text{ is a } B}$$

The conclusion is a logical consequence of the premises, we make explicit an information that was already in the premises.

#### 8.4.2. *Inductive inference*

$$\frac{P(a_1) \quad \vdots \quad P(a_n)}{\forall x P(x)}$$

The conclusion is not a logical consequence of the premises. We add information to the conclusion that was contained in the premises:

- The justification of inductive inference is realized by an accumulation of particular cases, thus, the conclusion is only likely.
- Some authors say: “an argumentation is *inductively strong* if the truth of the premises makes the truth of the conclusion *likely*”.

There are also paradoxes.

#### 8.4.3. *Hempel's paradox (1945)*

A plausible principle is that logically equivalent hypotheses are confirmed with the same degree by the same experimental data.

All ravens are black

$$\forall x[\text{raven}(x) \Rightarrow \text{black}(x)]$$

is logically equivalent to:

All objects that are not black are not ravens

$$\forall y[\neg \text{black}(y) \Rightarrow \neg \text{raven}(y)]$$

*By the plausible principle mentioned above, the observation of a white horse confirms the fact that all ravens are black!* □

DIGRESSION 8.2.– (on inductive inference). The presentation of inductive inference that has been made in section 8.4 is very concise and leads to the technique of inductive hypotheses generation proposed in section 8.5. In fact, it corresponds to the opinion of Aristotle who viewed induction as the act of getting *from the particular to the*

*general*. This point of view is a bit restrictive and somewhat dated. Since the 17th Century, and even more so since the axiomatization of probabilities by Kolmogorov (1933), inductive and probabilistic methods have been closely related<sup>8</sup>.

We may wonder what the differences are between induction and abduction (see section 8.3). The boundaries between these two great classes of non-deductive reasoning are not very precise. However, it should be noted that induction corresponds to *inferences in the context of uncertainty* (the uncertainty of an event denoted by a proposition  $P$  is defined as the probability of  $\neg P$ ), whereas abduction corresponds to a *theorization* (C.S. Peirce), i.e. to the imagination of a theory that is *explanatory* for the phenomena that are observed and, if possible, predictive of new discoveries. Probabilities provide clear foundations for induction but do not seem to be applicable to abduction.

Leibniz was the first philosopher of probability and the first to note that this theory could be used in an area of logic comparable to the theory of deduction. He incorporated probabilities into his theory of knowledge and anticipated what is known as inductive logic. He believed that the science of probability would become a new sort of logic, an idea that was taken up by J.M. Keynes<sup>9</sup>, H. Jeffrey, and R. Carnap in the 1920s. In this approach, it is accepted that there can be processes of non-deductive illustration, meaning that there can be good reasons to believe that a proposition  $P$  holds, *without*  $P$  being a logical consequence of other propositions. Carnap wanted to define an *objective* (and *syntactic*, i.e. exclusively related to the language that was used) measure of the degree up to which  $R$  is a reason for  $P$ .

By analogy with the notion of the proof of propositions that are logically necessary (deductive inferences are *necessary*)<sup>10</sup>, Leibniz proposed that the proof of a contingent proposition<sup>11</sup>  $P$  be an infinite sequence that asymptotically converges to  $P$ .

Inductive inference is *ampliative* (recall all the scientific discoveries that increase the field of what is known). Deductive induction is *explanatory*. From a classical point of view,<sup>12</sup> it is considered as not bringing any new knowledge: all the knowledge was *already* contained in the initial theory on which the inferences were carried out; the inference makes them explicit.

---

8 It is worth mentioning, for example, that, in his work, R. Carnap aimed at clarifying the concepts of degree of confirmation, inductive logic, and *probability*.

9 The economist who wrote a treatise on probability.

10 Mathematical induction (see section 3.3.2) is to be classified among the methods of deductive inference.

11 Contingent: likely to be or not to be, to occur or not to occur.

12 Which is criticized by some logic philosophers.

In other words, the information that is transmitted by deductive inference is void. A possible explanation for this choice is that  $P_1, P_2, \dots, P_n \models C$  iff  $P_1 \wedge P_2, \dots \wedge P_n \Rightarrow C$  is a tautology, and that it is admitted that the information transmitted by a wff (in a given language) is inversely proportional to the probability that the state of the world that it describes corresponds to reality<sup>13</sup>.

A more in-depth analysis of the concept of induction and the justification of inductive inference poses extremely difficult problems to philosophers of science. Several inductive logics (on which there is no consensus) have been proposed. J.M. Keynes characterized an inductive logic as a logic that studies “logical relations between two sets of propositions in cases where it is not possible to argue demonstratively from one to another”.

We have mentioned in section 8.4 what is called *induction by enumeration*, which must be distinguished from *induction by elimination*. *Induction by enumeration* permits us, starting from a sufficient number of facts, to obtain an *inductive consequence* (for example, all emeralds observed so far are green, therefore, all emeralds, including those not observed yet are green). *Induction by elimination* permits us, when enough alternate conclusions have been ruled out, to obtain an *inductive consequence*. This second form can be viewed as related to the advice by S. Holmes (see section 2.1.5.2) and to the usage of constraints (see section 9.2).

Some philosophers of science (in particular, K. Popper) defend the thesis that induction does not have its place in science, which they view as a *deductive* process based on hypotheses (theories) that scientists test using observable consequences. They can be *falsified* or *rejected* or *temporarily accepted*. These criticisms are mainly aimed at the forms of induction that have just been mentioned.

A form of induction that is frequently used is:

- the *observed* objects (i.e. the available evidence) that had property  $P$  also have property  $Q$ ;
- by assuming that object  $a$  (that has not been observed yet) has property  $P$ ;
- it is *likely* that  $a$  has property  $Q$ .

It is related to causal knowledge (laws of nature) and among the required conditions we can note:  $C$  is a cause of  $E$  if  $Prob(E | C) > Prob(\neg E | C)$ .

There exist different interpretations of the notion of probability<sup>14</sup>, among which the interpretation is known as *logical* or *inductive* (of which Carnap was the most

---

<sup>13</sup> This choice is consistent with information theory and with the property: if  $A \models B$  then  $P(B | A) = 1$ .

<sup>14</sup> Although the axiomatization by Kolmogorov has become canonical.



important defender), according to which every set of facts  $E$  uniquely determines the probability of a hypothesis  $H$  and in which the conditional probability  $Prob(X | Y)$  is considered as a quantitative generalization of the logical consequence between propositions  $Y$  and  $X$ . The key notion in inductive logic uses the notion of *confirmation* and provides a framework for induction: a piece of evidence  $E$  confirms a hypothesis  $H$  at the degree  $c(H, E) = \frac{m(H \wedge E)}{m(E)}$ , where  $m$  is a probability measure on the state of the world. The values  $c(H, E) = 1$  and  $c(H, E) = 0$  correspond to logical consequence and to incompatibility, respectively.  $c(H, E) > Prob(H)$ , (where  $Prob(H)$  is the *a priori* probability of  $H$ ) is a confirmation permitting us to learn from experience.

Proofs in such theories consist of the computation of the confirmation degree of pairs premises–conclusion.

Research on inductive logic is still carried out by researchers in philosophy of science, logic, and AI.

Of course, principles that relate (or differentiate<sup>15</sup>) the notions of deductive inference and of confirmation or verification have been searched for. One such example is the equivalence principle that was brought up in section 8.4.3 or the *implication principle*: if  $A$  confirms  $B$  then  $A$  confirms all logical consequences of  $B$ . But caution: if  $A$  is a logical consequence of  $C$ , then  $C$  does not necessarily confirm everything that is confirmed by  $A$ .

Imagine that the symptom denoted by proposition  $A$  is a confirmation that a patient has illness (denoted by proposition)  $M$ , but that the absence of the symptoms denoted by  $B$  and  $C$  allows us to reject the possibility that the patient has illness  $M$ . In other words,  $A \wedge \neg B \wedge \neg C$  does not confirm  $M$ , although  $A \wedge \neg B \wedge \neg C \models A$ . The notion of confirmation is therefore non-monotonic (see definition 2.6). For this reason, inductive inference and confirmation must respect the *principle of total evidence* that imposes that *all* relevant evidence should be taken into account in each induction.

In the same way that we require deductive inference rules be correct (see definition 3.12), we may wonder how to distinguish good inductions from bad instructions, or those that are reliable from those that are not. According to Hume, a formal justification cannot be expected, as a deductive justification is impossible and an inductive justification would lead to a circular argumentation. However, if it cannot

---

15 Jacques Bernoulli (18th Century) was one of the first to notice the difference between deductive logic that is used in situations of certain knowledge and inductive logic, which is necessary in the situations of uncertainty that are encountered in everyday life. He seems to be the first one to have actually related probability to logic: he provided a numerical measure of arguments and spoke of the “strength of a proof” or “degree of certainty”.

be justified, how can inductive inference be trusted? The answer that seems better adapted is that it is a probable inference. The law of large numbers that relates frequencies to probabilities can be viewed as setting the foundations of inductive inference (these laws are logical consequences of the axioms of probability theory and, although in empirical situations, additional hypotheses may be used, the inductive part of the reasoning depends on these laws that were established deductively).  $\square$

### 8.5. Generalization: the generation of inductive hypotheses

This is a fundamental problem. Generalization is one of the efficient ways of obtaining knowledge of the world: to build a taxonomy, in learning, in causal connections, etc.

**DEFINITION 8.3.**– (generalization). *A formula  $G$  is a generalization of a set of formulas  $F_i$  ( $1 \leq i \leq n$ ) iff  $G \models F_i$  for all  $i$ .*

For example, the discovery of a clause that subsumes another clause is a generalization. Subsumption is a weak form of logical consequence.

**DIGRESSION 8.3.**– Recent research in neurobiology and cognitive science has shown the importance for intelligent behaviors (in particular, logical reasoning) of *inhibition* mechanisms (of certain capabilities) that are used by the brain.

To make an analogy between the techniques presented above and the mechanisms that are used by human intelligence, the inhibited capability in generalization would be that of being able to distinguish details in the composition of sub-expressions.  $\square$

*Unification* is useful in *deductive* inference:

$$\begin{array}{ccc} P(a, f(y)) & & P(z, f(g(u))) \\ & \searrow & \swarrow \\ & P(a, f(g(u))) & \sigma : mgu \end{array}$$

the mgu  $\sigma$  is the less instantiated unifier and is thus the greatest instance (i.e. the most general, starting from the mgu, all other unifiers can be obtained).

*Generalization* is useful in *inductive* inference:

$$\begin{array}{ccc} & P(x, f(y)) & \\ \nearrow & & \nwarrow \\ P(a, f(c)) & & P(b, f(b)) \end{array}$$

Here, the useful generalization is the most instantiated generalization, and it is therefore the smallest one, i.e. the least general; every other one can be reduced to it by substitution.

We may say that  $P(x, f(y))$  explains  $P(a, f(c))$  and  $P(b, f(b))$ .

Why are we searching for the least general generalization (lgg)? Because, in principle, we can always generalize by replacing complex expressions by variables...but this way of proceeding is too general and useless: the specificities of the studied phenomenon are lost, together with the possibility of discovering an interesting law relating its different particular cases.

DEFINITION 8.4.– (generalization of terms, literals, and clauses). (*Note that this definition respects definitions 8.3 and 8.1, because in the case of literals, it is equivalent to requiring that  $G$  subsume  $F_i$ , and we know subsumption entails logical consequence (see exercise 8.1 (b)).*)

*The lgg of a set of expressions is a generalization  $G_{lgg}$  such that for every other generalization  $G$ ,  $G$  is more general than  $G_{lgg}$ , meaning that there exists a substitution  $\gamma$  such that  $G_{lgg} = \gamma G$ .*

*This concept can be applied to clauses. A clause  $C$  generalizes a clause  $D$  iff  $C$  subsumes  $D$ .*

EXAMPLE 8.11.– The lgg of:

$$F_1: P(f(g(\dots), k(y)), x, k(y))$$

and:

$$F_2: P(h(g(\dots), k(x)), x, k(x))$$

is:

$$F: P(y, x, k(z))$$

$$z \notin [Var(F_1) \cup Var(F_2)]$$

$$\sigma_1 F = F_1; \sigma_1 = \{z \leftarrow y, y \leftarrow f(g(\dots), k(y))\}$$

$$\sigma_2 F = F_2; \sigma_2 = \{z \leftarrow x, y \leftarrow h(g(\dots), k(x))\}$$

$F'$ :  $P(y, x, u)$  is also a generalization, but it is not lgg, as  $F = \gamma F'$  with  $\gamma = \{u \leftarrow k(z)\}$   $\square$

EXAMPLE 8.12.–  $C_1: \{Q(x) \vee P(g(h(\dots)), h(\dots))\}$

$$C_2: \{R(x) \vee P(g(k(\dots)), k(\dots))\}$$

the lgg of  $C_1$  and  $C_2$  is:

$$P(g(y), y)$$

$$y \notin [Var(C_1) \cup Var(C_2)].$$

□

**algorithm** lgg:

**input:** two terms, two literals, or two clauses  $w_1$  and  $w_2$

**output:**  $w = lgg(w_1, w_2)$

**begin**

**terms:**

- $w_1 : f(s_1, \dots, s_n) \quad w_2 : f(t_1, \dots, t_n)$   
 $w = f(lgg(s_1, t_1), \dots, lgg(s_n, t_n))$
- $w_1 : f(s_1, \dots, s_n) \quad w_2 : g(t_1, \dots, t_p)$   
**if**  $f \neq g$  and  $f(\dots), g(\dots)$  have not appeared before  
 % in particular  $f$  and  $g$  constants  
**then**  $w = X$  ( $X$ : new variable)  
**else**  $w = Y$  ( $Y$ : variable that has already been generalized  $f(\dots)$  or  $g(\dots)$ )

**literals:**

- $w_1 : P(s_1, \dots, s_n) \quad w_2 : \neg P(t_1, \dots, t_n)$   
 $w = \perp$
- $w_1 : P(s_1, \dots, s_n) \quad w_2 : P(t_1, \dots, t_n)$   
 $w = P(lgg(s_1, t_1), \dots, lgg(s_n, t_n))$

**clauses:**

- $w_1 : \{L_1, \dots, L_n\} \quad w_2 : \{M_1, \dots, M_p\}$   
 $w : \{lgg(L_i, M_j) \mid \exists L_i \in w_1, \exists M_j \in w_2, lgg(L_i, M_j) \neq \perp\}$

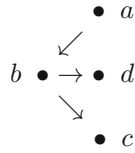
**end**

Figure 8.2. Generalization algorithm

EXAMPLE 8.13.– (explanation of concepts). An intelligent system has a knowledge base  $\mathbf{K}$ , represented by logical formulas (clauses), about a graph. This knowledge was obtained from different agents (these agents may have used different names for the same concept).

**K:**

- 1)  $edge(a, b)$
- 2)  $edge(b, c)$
- 3)  $edge(b, d)$
- 4)  $\forall x \forall y \forall z. \neg edge(x, z) \vee \neg elempath(z, y) \vee path(x, y)$



The system interrogates its environment to try to find an explanation of the concept of an **elempath** whose meaning it does not know, and obtains the following positive and negative examples:

**E**<sup>+</sup>:

5)  $path(a, c)$

6)  $path(a, d)$

**E**<sup>-</sup>:

7)  $\neg path(c, a)$

8)  $\neg path(d, c)$

– We verify that  $\mathbf{K} \cup \mathbf{E}^- \not\models \square$ .

All the consequences of  $\mathbf{K} \cup \mathbf{E}^-$  are:

9)  $\neg edge(c, z) \vee \neg elempath(z, a)$  (7, 1) – (4, 3)

10)  $\neg edge(d, z) \vee \neg elempath(z, c)$  (8, 1) – (4, 2)

– The set of all consequences of  $\mathbf{K}$  :

11)  $\neg elempath(b, y) \vee path(a, y)$  (1, 1) – (4, 1)

12)  $\neg elempath(c, y) \vee path(b, y)$  (2, 1) – (4, 1)

13)  $\neg elempath(d, y) \vee path(b, y)$  (3, 1) – (4, 1)

– The sets of consequences obtained are generally not finite. This example is a particular case in which the formulas do not contain any functional symbol.

– We search for the additional hypotheses that permit us to obtain  $\mathbf{E}^+$  as a set of logical consequences of the new knowledge base.

**H**<sub>1</sub>:

14)  $elempath(b, c)$

15)  $elempath(b, d)$

(14) and (15) permit us (resolution with (11)) to obtain (5) and (6)

Even better (from the point of view of the explanation):

**H<sub>2</sub>:**

We try to produce clauses that relate known concepts to the concept we are trying to explain (here, **elem**path) and from which the possible explanation that was already obtained (**H<sub>1</sub>**) can be deduced.

Candidates are:

- $\neg \text{edge}(a, b) \vee \text{elem}path(b, c)$  (resolution with (1) permits to obtain (14))
- $\neg \text{edge}(b, c) \vee \text{elem}path(b, c)$  (resolution with (2) permits to obtain (14))
- $\neg \text{edge}(b, d) \vee \text{elem}path(b, c)$  (resolution with (3) permits to obtain (14))
- $\neg \text{edge}(a, b) \vee \text{elem}path(b, d)$  (resolution with (1) permits to obtain (15))
- $\neg \text{edge}(b, c) \vee \text{elem}path(b, d)$  (resolution with (2) permits to obtain (15))
- $\neg \text{edge}(b, d) \vee \text{elem}path(b, d)$  (resolution with (3) permits to obtain (15))

The “best choice” seems to be:

- $\neg \text{edge}(b, c) \vee \text{elem}path(b, c)$
- $\neg \text{edge}(b, d) \vee \text{elem}path(b, d)$

Because it enables us, by generalization (see definition 8.4), to propose the explanation:

$$\boxed{\neg \text{edge}(x, y) \vee \text{elem}path(x, y)}$$

which in English reads as: for all  $x$  and all  $y$ , if there is an edge from  $x$  to  $y$ , then there is an elem

path from  $x$  to  $y$ , or, simply put, every edge is an elempath.

### 8.5.1. Generalization from examples and counter examples

In the generalization of terms, counter examples could also be used.

EXAMPLE 8.14.– (taking examples and counter examples into account). We assume given a signature (essential hypothesis) and **examples:**  $f(b, a)$ ,  $f(a, b)$ , and  $f(a, c)$

and **counter examples:**  $f(a, a)$  and  $f(c, c)$

A generalization would be:

$$f(x, y) \setminus \{f(u, u)\}$$

meaning: set of constant instances of  $f(x, y)$  such that  $x \neq y$  (note that a constraint has been introduced).  $\square$

REMARK 8.7.– The given representation is an *implicit* representation of the set of denoted terms, but does not give the *form* (structure) of these terms.

At least two questions arise:

- does there exist a *finite* explicit representation of denoted terms?
- is this representation computable?

EXAMPLE 8.15.– (of an explicit representation). If we consider the set of terms:

$$f(x, y) \setminus \{f(a, u) \vee f(u, a)\}$$

and we want to give an explicit representation for them, it is necessary to fix the signature of the corresponding Herbrand signature.

$$\Sigma = \{a, b, f^{(2)}\}$$

The set of terms

$$\{a, b, f(a, a), f(a, b), f(b, a), f(b, b), f(a, f(a, a)), \dots\}$$

has the following finite explicit representation:

$$\{f(b, b) \vee f(f(x, y), b) \vee f(b, f(x, y)) \vee f(f(x, y), f(x', y'))\}. \quad \square$$

EXAMPLE 8.16.– (where there is no explicit representation). The set of terms:

$$f(x, y) \setminus \{f(u, u)\}$$

does not have any finite representation on (for example) the signature:

$$\Sigma = \{a, g^{(1)}\}$$

The set of denoted terms is a proper subset (meaning that it does not contain the framed terms in (\*)) of the set of terms on signature  $\Sigma$ :

$$\{a, g(a), g^2(a), \dots, g^n(a), \dots\} \quad n \in \mathbb{N}$$

$$(*) \left\{ \boxed{f(a, a)}, f(a, g(a)), f(g(a), a), \boxed{f(g(a), g(a))}, f(g^2(a), a), \dots \right\}. \quad \square$$

EXAMPLE 8.17.– (where the set of terms is empty).  $f(x, y) \setminus \{f(u, u) \vee f(f(u, v), w) \vee f(w, f(u, v)) \vee f(a, f(f(u, v), w))\}$

on signature  $\Sigma = \{a, f^{(2)}\}$

denotes  $\emptyset$ . □

In the case in which an explicit representation can be provided, this information can be used for a more precise treatment of negation as failure.

EXAMPLE 8.18.– (of a “more detailed” treatment of NAF). The program

$pp(f(a)) \rightarrow ;$

$qq(f(X)) \rightarrow \text{not}(pp(X)) ;$

together with the question

$qq(Z) ;$

will answer no.

Actually, we should obtain:

$Z = f(a) \vee f(f(f(Y)))$

(the complement of  $pp(f(a))$  on the implicit signature  $\Sigma = \{a, f^{(1)}\}$  is  $pp(a) \vee pp(f(f(y)))$ , i.e.  $x = a \vee x = f(f(y))$ ).  $\square$



## Chapter 9

# Problem Specification in Logical Languages

### 9.1. Equality

Equality is an extremely important<sup>1</sup> predicate (relation), in particular, in mathematics, that has a meaning in *every* universe of discourse, which is not the case for other predicates (for example, if  $P(x, y)$  has the intended meaning “ $x$  loves  $y$ ”, it would have no meaning if the universe of discourse was, say,  $\mathbb{N}$ ).

It seems natural to want to treat it using logics that we have already studied.

Imagine we have to prove the validity (or non-validity) of the equivalence:

$$(*) \quad \underbrace{x = y}_P \wedge \underbrace{x = z}_Q \Leftrightarrow \underbrace{x = y}_P \wedge \underbrace{y = z}_R$$

If propositional logic (PL) is used, using the names mentioned, we get  $P \wedge Q \Leftrightarrow P \wedge R$ , which is obviously a non-valid formula, but our experience with  $=$  tells us that formula (\*) is valid, and we want to classify it as such.

FOL would also fail (without any additional axioms) to capture the characteristics of equality.

Equality has particular properties that we must make explicit.

There are formulas such as:

---

<sup>1</sup> A philosopher of logic (W.V.O. Quine) wrote “There is still no entity, no set, nothing without an identity”.

$$\exists y(R(y) \wedge P(y)) \Rightarrow \exists yR(y)$$

that are valid when  $R$  is replaced by  $x = y$  or by any other predicate  $Q(x, y)$ .

But the validity of the formula

$$P(x) \wedge (x = y) \Rightarrow P(y)$$

depends on the *semantics* of  $=$ .

### 9.1.1. When is it used?

EXAMPLE 9.1.– There exists a unique element with property P:

$$\text{i) } \exists x(P(x) \wedge \forall y(P(y) \Rightarrow (x = y)))$$

or:

$$\text{ii) } \exists xP(x) \wedge \forall x\forall y(P(x) \wedge P(y) \Rightarrow (x = y))$$

(i) and (ii) are often abbreviated

$$\exists!xP(x)$$

There are at least two objects with property P:

$$\text{iii) } \exists x\exists y(P(x) \wedge P(y) \wedge (x \neq y))$$

There are at most two objects with property P:

$$\text{iv) } \forall x\forall y\forall z(P(x) \wedge P(y) \wedge P(z) \Rightarrow (x = y) \vee (y = z) \vee (x = z))$$

There are exactly two objects with property P:

$$\text{v) } \exists x\exists y(P(x) \wedge P(y) \wedge (x \neq y) \wedge \forall z(P(z) \Rightarrow (x = z) \vee (y = z)))$$

It is clear that:

$$\text{(iii) } \wedge \text{(iv) } \Leftrightarrow \text{(v)}$$

□

### 9.1.2. Some questions about equality

- 1) Why is equality needed?
- 2) What is equality?
- 3) How can we reason with equality?

### 9.1.3. Why is equality needed?

Since:

- everything is identical to itself and to nothing else.
- identifying a thing to itself is trivial and identifying it to something else is false.

*Where does the need for and usefulness of equality come from?*

From the fact that we allow different names for the same object.

(It is not the *names* that are identical, but the *named objects*).

### 9.1.4. What is equality?

Equality is an equivalence relation with the replacement or substitution property.

To formalize the equality between two objects, Leibniz expressed (Leibniz's law)<sup>2</sup>

$$\forall x \forall y ((x = y) \Leftrightarrow \forall P (P(x) \Leftrightarrow P(y)))$$

Leibniz's law is sometimes presented as the conjunction of the two following laws (the first is the common law in mathematics):

$$\forall x \forall y ((x = y) \Rightarrow \forall P (P(x) \Leftrightarrow P(y)))$$

**Substitution principle** (*indistinguishability of identity*)

$$\forall x \forall y (\forall P (P(x) \Leftrightarrow P(y)) \Rightarrow (x = y))$$

**Indiscernibility principle** (*identity of indiscernibles*)

(Leibniz's law is not a wff of FOL).

Axiomatization of equality:

- 1)  $\forall x (x = x)$
- 2)  $\forall x \forall y (x = y) \Rightarrow (y = x)$
- 3)  $\forall x \forall y \forall z ((x = y) \wedge (y = z)) \Rightarrow (x = z)$
- 4)  $\forall f \forall x \forall y (x = y) \Rightarrow (f(\bar{u}, x, \bar{z}) = f(\bar{u}, y, \bar{z}))$
- 5)  $\forall P \forall x \forall y ((x = y) \wedge P(\bar{u}, x, \bar{z})) \Rightarrow P(\bar{u}, y, \bar{z})$

---

<sup>2</sup> Some authors have noted that it can be simplified:  $x = y \Leftrightarrow \forall P (P(x) \Rightarrow P(y))$ .

where  $\bar{u}$  and  $\bar{z}$ , respectively, denote  $u_1, \dots, u_m$  and  $z_1, \dots, z_n$  ( $0 \leq m, n$ ) (Of course, here we assume that  $f$  and  $P$  are of arity  $m + n + 1$ ).

Note that (4) and (5) are not wffs of FOL (which does not allow for the quantification of function or predicate symbols). Therefore, equality cannot be defined in FOL, and of course, it cannot be defined in PL either.

*However equality can be treated in FOL.*

When treating formulas that contain equalities in FOL, the latter can be axiomatized in this particular setting simply by noticing that in every wff of FOL, there are finitely many predicate and function symbols. Formulas (4) and (5) can be written for each of these symbols.

This means that we need to add:

$$\sum_{i=1}^{n_f} ar_{f_i} + \sum_{j=1}^{n_p} ar_{p_j} \text{ formulas,}$$

where:

$n_f$ : number of functional symbols in the formula;

$n_p$ : number of predicate symbols in the formula;

$ar_{f_i}$ : arity of function  $f_i$ ; and

$ar_{p_j}$ : arity of predicate  $p_j$ .

REMARK 9.1.– Equality can be axiomatized with (1), (4), and (5) (see exercises 9.1 and 9.2).  $\square$

EXAMPLE 9.2.– We want to reason on the set of clauses  $S$  below:

- 1)  $P(f(a), g(e))$
- 2)  $\neg P(f(x), g(x))$
- 3)  $f(a) = f(b)$
- 4)  $b = c$
- 5)  $e = c$

We add the clauses:

(recall that:  $P \wedge Q \Rightarrow R \Leftrightarrow \neg P \vee \neg Q \vee R$ )

- 6)  $x = x$
- 7)  $\neg(x = y) \vee (y = x)$

- 8)  $\neg(x = y) \vee \neg(y = z) \vee (x = z)$
- 9)  $\neg(x = x') \vee \neg P(x, y) \vee P(x', y)$
- 10)  $\neg(y = y') \vee \neg P(x, y) \vee P(x, y')$
- 11)  $\neg(x = x') \vee f(x) = f(x')$
- 12)  $\neg(x = x') \vee g(x) = g(x')$

Of course, we could have written equality as an ordinary predicate with the prefix notation:  $E(\dots, \dots)$ , but we prefer the standard mathematical notation.  $\square$

REMARK 9.2.– Substitution seems natural, in fact it is natural for extensional languages. This is not the case for *non*-extensional languages.

Consider the following sentence:

i) Michael knows that the sum of the first  $n$  odd numbers is  $1 + 3 + 5 + \dots + (2 \times n + 1)$ .

The sentence:

ii) If Michael knows that the sum of the first  $n$  odd numbers is  $1 + 3 + 5 + \dots + (2 \times n + 1)$ , then Michael knows that the sum of the first  $n$  numbers is  $\underbrace{1 + 3 + 5 + \dots + (2 \times n + 1)}$

is trivially true (it is actually a tautology).

Furthermore,

iii)  $1 + 3 + 5 + \dots + (2 \times n + 1) = n^2$ .

Using (iii) and *substituting*  $\underbrace{1 + 3 + 5 + \dots + (2 \times n + 1)}$  by  $n^2$  in (ii) leads to:

iv) If Michael knows that the sum of the first  $n$  odd numbers is  $1 + 3 + 5 + \dots + (2 \times n + 1)$ , then Michael knows that the sum of the first  $n$  odd numbers is  $n^2$ .

... and what used to be a tautology does not remain so, because Michael may not be aware of property (iii).  $\square$

### 9.1.5. How to reason with equality?

It is possible to handle equality by resolution, by axiomatizing it (see the following section) or using the paramodulation rule (see definition 9.1).

The following example shows that it is sometimes possible not to bother about equality in the statement of some problems (here, a theorem). The way of proceeding is the way used in logic programming (see Chapter 6), where a key technique is the *naming* technique.

### 9.1.6. Specification without equality

EXAMPLE 9.3.– (a theorem in group theory, without =). We prove the following theorem (see also example 9.13) by resolution:

If in a group  $G$  we have:  $\forall x.x^2 = x$ , then  $G$  is commutative.

The associativity of an operation  $\circ$  is expressed by:

$$\forall x \forall y \forall z. \underbrace{(x \circ y)}_w \circ z = x \circ \underbrace{(y \circ z)}_v$$

By naming as indicated by the braces, we can express the associativity of  $\circ$  using the predicate:

$P(x, y, z)$ : the composition of  $x$  and  $y$  yields  $z$ .

We thus express with clauses:  $x \circ v = w$  iff  $u \circ z = w$

$\forall x \forall y \forall z \forall u \forall v \forall w. P(x, y, u) \wedge P(y, z, v) \wedge P(u, z, w) \Rightarrow P(x, v, w)$  (if)

$\forall x \forall y \forall z \forall u \forall v \forall w. P(x, y, u) \wedge P(y, z, v) \wedge P(x, v, w) \Rightarrow P(u, z, w)$  (only if)

Identity will be denoted by the constant  $e$  and the inverse of an element  $x$  by  $i(x)$ .

The conclusion of the theorem (i.e. commutativity):

$$\forall x \forall y \forall z. P(x, y, z) \Rightarrow P(y, x, z)$$

and its negation:

$$\neg(\forall x \forall y \forall z. P(x, y, z) \Rightarrow P(y, x, z))$$

$$\exists x \exists y \exists z \neg(P(x, y, z) \Rightarrow P(y, x, z))$$

after Skolemization, we obtain two clauses:

$$P(a, b, c) \wedge \neg P(b, a, c)$$

Proving the theorem therefore boils down to deducing 2 from clauses (1) to (9) below:

- 1)  $\neg P(x, y, u) \vee \neg P(y, z, v) \vee \neg P(u, z, w) \vee P(x, v, w)$
- 2)  $\neg P(x, y, u) \vee \neg P(y, z, v) \vee \neg P(x, v, w) \vee P(u, z, w)$
- 3)  $P(e, x, x)$
- 4)  $P(x, e, x)$
- 5)  $P(i(x), x, e)$
- 6)  $P(x, i(x), e)$
- 7)  $P(x, x, e)$
- 8)  $P(a, b, c)$
- 9)  $\neg P(b, a, c)$

We shall use the *hyperresolution* rule without having defined it formally; it corresponds to applying several resolution steps in one single step.

- 10)  $P(c, b, a)$  (2, 1) – (8, 1); (2, 2) – (7, 1); (2, 3) – (4, 1)

$$\{x \leftarrow a, y \leftarrow b, z \leftarrow b, u \leftarrow c, v \leftarrow e, w \leftarrow a\}$$

- 11)  $P(c, a, b)$  (1, 1) – (7, 1); (1, 2) – (10, 1); (1, 3) – (3, 1)

$$\{x \leftarrow c, y \leftarrow c, z \leftarrow b, u \leftarrow e, v \leftarrow a, w \leftarrow b\}$$

- 12)  $P(b, a, c)$  (2, 1) – (11, 1); (2, 2) – (7, 1); (2, 3) – (4, 1)

$$\{x \leftarrow c, y \leftarrow a, z \leftarrow a, u \leftarrow b, v \leftarrow e, w \leftarrow c\}$$

- 13)  $\square$  (12, 1) – (9, 1). □

### 9.1.7. Axiomatization of equality

Once it has been axiomatized, equality is a predicate like any other predicate, and its definition can be incorporated by adding formulas to the problems that use it, and then using the method of semantic tableaux or of resolution. The goal is to *extend* these methods. Of course, here, *extension* means incorporating rules that take the properties of = into account into these methods.

### 9.1.8. Adding the definition of = and using the resolution method

EXAMPLE 9.4.– We give a linear refutation by resolution, for the set of clauses  $S$  of example 9.2:

- |  |  |
|--|--|
| 13) $\neg(f(a) = x') \vee P(x', g(e))$ | (1,1)-(9,2) $\{x \leftarrow f(a), y \leftarrow g(e)\}$   |
| 14) $P(f(b), g(e))$                    | (13,1)-(3,1) $\{x' \leftarrow f(b)\}$                    |
| 15) $P(f(b), y') \vee \neg(g(e) = y')$ | (14,1)-(10,2) $\{x \leftarrow f(b), y \leftarrow g(e)\}$ |
| 16) $\neg(g(e) = g(b))$                | (15,1)-(2,1) $\{x \leftarrow b, y' \leftarrow g(b)\}$    |
| 17) $\neg(e = b)$                      | (16,1)-(12,2) $\{x \leftarrow e, x' \leftarrow b\}$      |
| 18) $\neg(e = y) \vee \neg(y = b)$     | (17,1)-(8,3) $\{x \leftarrow e, z \leftarrow b\}$        |
| 19) $\neg(c = b)$                      | (18,1)-(5,1) $\{y \leftarrow c\}$                        |
| 20) $\neg(b = c)$                      | (19,1)-(7,2) $\{y \leftarrow c, x \leftarrow b\}$        |
| 21) $\square$                          | (20,1)-(4,1)   |

□

REMARK 9.3.– Equality is not handled by Prolog, in which it is possible to use the predicate `eq` that can be evaluated (see exercise 6.10).

To convince oneself that the predicate does not correspond to equality, it suffices to use the program (aa, bb, and cc denote constants):

```
eq(aa, bb) ->;
```

```
eq(bb, cc) ->;
```

and to ask the question:

```
eq(aa, cc);
```

The answer will be “false”, which does not correspond to the answer we would get with the usual notion of equality (that was formally defined in section 9.1.4). □

REMARK 9.4.– Reasoning automatically with equality can be very difficult. If one is interested in reasoning only on finite domains, problems can often be formulated in such a way to avoid its occurrence, as shown in the following example. □

EXAMPLE 9.5.– We assume that there is a set of candidates to a set of available jobs, and we want to express that there cannot be two candidates who get the same job.

The wff of FOL with equality (FOLE or  $\text{FOL}^=$ ) correctly formalizes the statement of the problem.

$$(*) \forall x \forall y \forall z \forall w (R(x, y) \wedge R(z, w) \wedge x \neq z \Rightarrow y \neq w)$$

where  $R(x, y)$  means: job  $y$  is given to candidate  $x$ .

Once the number of candidates has been fixed, say Alice (a), Bob (b), Carrie (c), and Daniel (d), we can express the statement (\*) (with the objective of not having to handle equality) with the six (i.e.  $C_4^2$ ) following clauses:



$$\forall x \neg (R(a, x) \wedge R(b, x))$$

$$\forall x \neg (R(a, x) \wedge R(c, x))$$

$$\forall x \neg (R(a, x) \wedge R(d, x))$$

$$\forall x \neg (R(b, x) \wedge R(c, x))$$

$$\forall x \neg (R(b, x) \wedge R(d, x))$$

$$\forall x \neg (R(c, x) \wedge R(d, x))$$

□

### 9.1.9. By adding specialized rules to the method of semantic tableaux

The following assertions are implicitly accepted in general, but deserve to be restated:

- names (constants or terms without variables) denote an object that exists. This assumption is necessary to be able to adopt the (quite natural!) point of view admitting that  $a \neq a$  (i.e.  $\neg(a = a)$ ) is contradictory;

- function symbols denote *total* functions, i.e. if  $f$  is an arbitrary function symbol and  $a$  is a constant, then  $f(a) = b$ , where  $b$  is a constant denoting an object that exists, as recalled above.

$R_1^-$ :

If a branch  $\mathcal{B}$  contains  $\neg(a = a)$  or  $\neg(f^n(\bar{t}) = f^n(\bar{t}))$ , where  $\bar{t}$  is an  $n$ -tuple of closed terms: put a  $\times$  in  $\mathcal{B}$ .

$R_2^-$ :

For every constant (or term denoting a constant) and every closed formula, use substitution (axioms 4 and 5 in the axiomatization of equality).

REMARK 9.5.– To avoid strategies that would *artificially* introduce non-termination, we will require that the closed formulas produced by substitution do not already occur in the branch (the same remark as for universal quantifiers). □

Do we need to add rules that handle symmetry and transitivity?

If it is possible to prove the validity of these axioms using  $R_1^-$  and  $R_2^-$ , then we will have answered “no” to the question and we will be done with the extension of the method of semantic tableaux incorporating equality<sup>3</sup>.

---

<sup>3</sup> Of course, once the properties of symmetry and transitivity have been proved, they can be used as if they were axioms.

EXERCISE 9.1.– (symmetry of  $=$ ). Prove, using the method of semantic tableaux extended with  $R_1^=$  and  $R_2^=$ , the validity of:

$$\forall x \forall y (x = y \Rightarrow y = x). \quad \square$$

EXERCISE 9.2.– (transitivity of  $=$ ). Use the method of semantic tableaux extended with  $R_1^=$  and  $R_2^=$  to prove the validity of:

$$\forall x \forall y \forall z [(x = y) \wedge (y = z) \Rightarrow (x = z)]. \quad \square$$

We can therefore extend the method of semantic tableaux with  $R_1^=$  and  $R_2^=$ , and it can be applied to FOLE (FOL $^=$ ).

We easily obtain the procedure to do so (the added lines are preceded by  $=$ ).

EXERCISE 9.3.– Use the method of semantic tableaux with equality to prove the assertion of example 9.1, i.e. (iii)  $\wedge$  (iv)  $\Leftrightarrow$  (v).  $\square$

### 9.1.10. *By adding specialized rules to resolution*

#### 9.1.10.1. *Paramodulation and demodulation*

These rules were introduced to handle reasoning on clauses that contain literals involving equality.

We first study the paramodulation rule. It permits us in one step to combine the following operations:

- instantiation: replacement of variables by terms;
- replacement of equals by equals.

For example, if we have the two clauses (1) and (2) below:

$$1) f(x) = g(a) \quad \% \text{ recall: for all } x$$

$$2) f(b) = c$$

from (1) we may deduce:

$$3) f(b) = g(a)$$

and, using (3) and replacing equals by equals in (2), we obtain:

$$2) g(a) = c$$

Such deductions are performed in one step with the paramodulation rule.

```

procedure SEMANTIC TABLEAUX WITH EQUALITY (FOL=)
% constructs the tree
input: a finite set of wffs of FOL=  $\mathcal{F} = \{f_1, \dots, f_n\}$ 
output: (if the procedure halts) models of  $\mathcal{F}$  or  $\mathcal{F}$  contradictory
% it may not halt (FOL= undecidable)
begin
    root of the tree  $\leftarrow \mathcal{F}$ 
    while  $\mathcal{F} \neq \emptyset$  and (there are open branches)
    do
        if a branch  $\mathcal{B}$  contains 2 compl. c-literals close  $\mathcal{B}$ 
        (i.e. put a  $\times$ )
        % the corresponding model is not viable
        (=) apply  $R_1^-$ 
        choose  $f_i \in \mathcal{F}$ 
        if  $f_i$  of the form  $[\neg] (F [< connective > G])$ 
        % [...] : possibly non-existent
        then apply rules  $\alpha$  and  $\beta$ : we obtain  $f_i^j$  ( $1 \leq j \leq 2$ )
        Graft  $f_i^j$  ( $1 \leq j \leq 2$ ) in all the
        open branches going through the node labelled by  $f_i$ 
         $\mathcal{F} \leftarrow \mathcal{F} \setminus \{f_i\}$ 
        else if  $f_i$  of the form  $\exists xG$ 
        then apply rule  $\delta$  and do  $\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\}) \cup G[x \leftarrow t]$ 
        % we use existential formulas only once
        else if  $f_i$  of the form  $\forall xG$  then apply rule  $\gamma$  and do
         $\mathcal{F} \leftarrow \mathcal{F} \cup G[x \leftarrow t]$ 
        % we keep  $f_i$  indefinitely
        (=) apply  $R_2^-$ 
    enddo

    if closed tree
    then return  $\mathcal{F}$  contradictory
    else return the sets of literals of the open branches
end
    
```

Figure 9.1. Procedure (FOL<sup>=</sup>)

DEFINITION 9.1.– (paramodulation rule). We note:

$L[t]_u$ : term  $t$  occurs at position  $u$  in literal (or term)  $L$ .

$L[t]$ : term  $t$  occurs at an unspecified position in literal (or term)  $L$ .

$L[u \leftarrow t]$ : the literal (or term) obtained by replacing the term occurring at position  $u$  by  $t$ .

Consider two clauses  $C$  and  $D$ :

$$C: s = t \vee C_1$$

$$D: L \vee D_1$$

where:

$L$ : a literal (in particular, an equality or the negation of an equality).

$C_1, D_1$ : clauses.

If  $L[r]_u$  and  $r \doteq s$  admit a solution  $\sigma$ , then the clause

$$\sigma(C_1 \vee L[u \leftarrow t] \vee D_1)$$

is a paramodulant of  $C$  in  $D$ .

EXAMPLE 9.6.–  $C: \underline{f(x, g(x))} = e \vee Q(x)$

$$D: P(y, \underline{f(g(y), z)}, z) \vee R(z)$$

$$\sigma = \{x \leftarrow g(y), z \leftarrow g(g(y))\}$$

A paramodulant of  $C$  in  $D$  is the clause:

$$P(y, e, g(g(y))) \vee Q(g(y)) \vee R(g(g(y))). \quad \square$$

EXAMPLE 9.7.– Prove, using the paramodulation and resolution rules, that the set containing the three clauses below is E-unsatisfiable (i.e. if the predicate =, denoted here in infix mode as usual, represents equality).

$$1) P(x, x, f(a), f(b))$$

$$2) \neg P(f(a), f(b), x, x)$$

$$3) a = b$$

$$4) P(x, x, f(a), f(a)) \quad \text{paramodulation of (3) in (1)}$$

$$5) \neg P(f(a), f(a), x, x) \quad \text{paramodulation of (3) in (2)}$$

$$6) \square \quad \text{resolution (4,1)-(5,1) } \{x_4 \leftarrow f(a), x_5 \leftarrow f(a)\}. \quad \square$$

REMARK 9.6.– (reflexivity must not be forgotten). Although it is not necessary in this example, when reasoning with equality, it is necessary to add the clause:

$$x = x \quad \% \text{ recall: for all } x$$

in order not to lose refutational completeness.

Otherwise, it is impossible to prove by resolution and paramodulation that the set of clauses  $\{a \neq a\}$  is contradictory.

If we add  $x = x$ , then, by resolution, we obtain  $\square$  with  $\{x \leftarrow a\}$ .  $\square$

EXAMPLE 9.8.– Prove, using the paramodulation and resolution rules, that the set consisting of the four clauses below is E-unsatisfiable (i.e.  $S$  is unsatisfiable when  $=$  is interpreted as the usual equality, see section 9.1.4).

- 1)  $P(a) \vee a = b$
- 2)  $P(b)$
- 3)  $\neg P(a) \vee Q(c)$
- 4)  $\neg P(a) \vee \neg Q(c)$
- 5)  $P(a)$  paramodulation of (1) in (2)
- 6)  $Q(c)$  resolution (5,1)-(3,1)
- 7)  $\neg P(a)$  resolution (6,1)-(4,2)
- 8)  $\square$  resolution (7,1)-(5,1).  $\square$

Demodulation or rewriting or reduction uses equalities to replace terms by “simpler” terms, we thus need complexity measures, and if, for example, we consider brackets on the right as simpler, we *orient* the equality  $(x + y) + z = x + (y + z)$  as  $(x + y) + z \rightarrow x + (y + z)$ .

The goal is to keep information in its simplest form, if possible a canonical form. For example,  $a + 0 \rightarrow a$ . This, thus, corresponds to algebraic simplification (symbolic computation).

EXAMPLE 9.9.– a) When considering the unit clause:

$$f(f(x)) = g(x)$$

oriented into:

$$f(f(x)) \rightarrow g(x)$$

the clause:

$$P(f(f(a)), b)$$

is demodulated into

$$P(g(a), b) \text{ and removed.}$$

b) When considering the unit clause:

$$x + 0 = x$$

oriented into:

$$x + 0 \rightarrow x$$

the clause:

$$P((a + 0) + b, c)$$

is demodulated into

$$P(a + b, c) \text{ and removed.} \quad \square$$

DEFINITION 9.2.– (demodulation rule). *We write:*

$L[t]_u$ : *term  $t$  occurs at position  $u$  in literal (or term)  $L$ .*

$L[u \leftarrow t]$ : *is the literal (or term) obtained by replacing the term at position  $u$  by  $t$ .*

*Consider the two clauses  $C$  (unit, with equality as a predicate) and  $D$ :*

$C$ :  $s = t$  *the demodulator*

$D$ :  $L \vee D_1$

*where:*

$L$ : *literal (in particular, an equality or the negation of an equality).*

$D_1$ : *a clause.*

*If  $L[r]_u$  and there exists a substitution  $\sigma$  such that  $\sigma s = r$  (matching instead of unification),*

*then the clause*

$\sigma(L[u \leftarrow t] \vee D_1)$  *that replaces  $D$*

*was obtained by demodulating  $D$  with  $C$ .*

EXAMPLE 9.10.– (is demodulation complete?). The answer is no, as shown in the following example:

$$S = \{P(a), \neg P(b), a = f(c), b = f(c)\}$$

$S$  is E-unsatisfiable, but a natural complexity measure would orient the equalities  $f(c) \rightarrow a$ ;  $f(c) \rightarrow b$  and it would be impossible to derive a contradiction.  $\square$

EXAMPLE 9.11.– (what if = occurs in non-unit clauses?). The E-unsatisfiable set of clauses (1) to (4) below shows that restricting the usage of = to its occurrences in unit clauses is too strong a requirement (meaning that unsatisfiable sets of clauses are not detected as so, and completeness is lost).

- 1)  $(c = d) \vee \neg Q(c)$
- 2)  $g(c) \neq g(d) \vee \neg Q(c)$
- 3)  $(a = b) \vee Q(c)$
- 4)  $g(a) \neq g(b) \vee Q(c)$

$S$  is E-unsatisfiable, a result that is easily proved by resolution, by axiomatizing equality with the technique that was already presented. We add:

- 5)  $\neg(a = b) \vee (g(a) = g(b))$
- 6)  $\neg(c = d) \vee (g(c) = g(d))$

and we obtain:

- 7)  $g(a) = g(b) \vee Q(c)$  (3, 1) – (5, 1)
- 8)  $g(c) = g(d) \vee \neg Q(c)$  (1, 1) – (6, 1)
- 9)  $Q(c)$  (4, 1) – (7, 1)
- 10)  $\neg Q(c)$  (2, 1) – (8, 1)
- 11)  $\square$  (9, 1) – (10, 1).  $\square$

EXAMPLE 9.12.– (demodulation: a canonical procedure?). The answer is no. Consider the clause:

$$(*) Q(f(f(a, b), c))$$

and the set of demodulators:

- 1)  $f(a, b) \rightarrow d$
- 2)  $f(b, c) \rightarrow e$
- 3)  $f(f(x, y), z) \rightarrow f(x, f(y, z))$

By using (1), (\*) can be demodulated into:

Demodulation	Paramodulation
Uses substitution =	idem
matching	unification
= in unit clause	not necessarily
demodulated clause removed	no removal

**Figure 9.2.** Paramodulation versus demodulation

$$i) \quad Q(f(d, c))$$

By using (3), (\*) can be demodulated into:

$$Q(f(a, f(b, c)))$$

and (2) into:

$$ii) \quad Q(f(a, e))$$

(i) and (ii) are two equivalent clauses, but one cannot be reduced to the other by demodulation.  $\square$

Figure 9.2 summarizes the differences between demodulation and paramodulation.

EXAMPLE 9.13.– (a theorem in group theory). If in a group  $G$ ,  $\forall x.x^2 = e$ , then  $G$  is commutative.

A human would probably give a proof similar to the following (of course,  $\circ$  denotes the operation in  $G$ ):

$$\underbrace{(y \circ z)}_x \circ \underbrace{(y \circ z)}_x = e$$

$$\underbrace{y \circ y}_e \circ (z \circ y \circ z) = \underbrace{y \circ e}_y$$

$$z \circ y \circ z = y$$

$$\underbrace{z \circ z}_e \circ y \circ z = z \circ y$$

$$y \circ z = z \circ y$$

Another human proof (using another strategy) could be:



$$1) z = x \circ y$$

$$z \circ y = (x \circ y) \circ y = x \circ (y \circ y) = x \circ e = x$$

$$z \circ (z \circ y) = (z \circ z) \circ y = e \circ y = z \circ x$$

$$y = z \circ x$$

$$2) y \circ x = (z \circ x) \circ x = z \circ (x \circ x) = z \circ e = z$$

hence ((1) and (2))

$$x \circ y = y \circ x$$

To give a proof by paramodulation, demodulation, and resolution, the group axioms (which are implicit in the proof above) need to be written in clausal form, and the conclusion must be negated:

$$1) f(e, x) = x \quad \% f(x, y): x \circ y$$

$$2) f(x, e) = x$$

$$3) f(g(x), x) = e \quad \% g(x): x^{-1}$$

$$4) f(x, g(x)) = e$$

$$5) f(f(x, y), z) = f(x, f(y, z)) \quad \% \circ \text{ is associative}$$

$$6) f(x, x) = e$$

$$7) f(a, b) \neq f(b, a) \quad \% \text{ neg. of conclusion and Skolemization}$$

$$(\neg[\forall x \forall y. f(x, y) = f(y, x)] \text{ equiv. } \exists x \exists y. \neg[f(x, y) = f(y, x)] \text{ skol.:}$$

$$\neg(f(a, b) = f(b, a)))$$

$$8) x = x \quad \% \text{ not used in this example}$$

Paramodulation of (5) into (6):

$$5) \underline{f(f(x_5, y_5), z_5)} = f(x_5, f(y_5, z_5))$$

$$6) \underline{f(x_6, x_6)} = e$$

$$\sigma = \{x_6 \leftarrow f(x_5, y_5), z_5 \leftarrow f(x_5, y_5)\}$$

We obtain (after renaming the variables):

$$9) f(x_9, f(y_9, f(x_9, y_9))) = e$$

Paramodulation of (6) into (5):

$$6) \underline{f(x_6, x_6)} = e$$

$$5) \underline{f(f(x_5, y_5), z_5)} = f(x_5, f(y_5, z_5))$$

$$\sigma = \{x_6 \leftarrow x_5, y_5 \leftarrow x_5\}$$

we obtain:

$$\underline{f(e, z_5)} = f(x_5, f(x_5, z_5)),$$

Demodulation with (1):

$$1) \underline{f(e, x_1)} = x_1$$

$$\sigma = \{x_1 \leftarrow z_5\}$$

We obtain (after renaming the variables):

$$10) f(x_{10}, f(x_{10}, z_{10})) = z_{10}$$

Paramodulation of (9) into (10):

$$9) \underline{f(x_9, f(y_9, f(x_9, y_9)))} = e$$

$$10) f(x_{10}, \underline{f(x_{10}, z_{10})}) = z_{10}$$

$$\sigma = \{x_{10} \leftarrow x_9, z_{10} \leftarrow f(y_9, f(x_9, y_9))\}$$

The paramodulant is:

$$\underline{f(x_9, e)} = f(y_9, f(x_9, y_9))$$

Demodulation with (2):

$$2) \underline{f(x_2, e)} = x_2$$

$$\sigma = \{x_2 \leftarrow x_9\}$$

We obtain (after renaming the variables):

$$11) f(y_{11}, f(x_{11}, y_{11})) = x_{11}$$

Paramodulation of (11) into (10):

$$11) \underline{f(y_{11}, f(x_{11}, y_{11}))} = x_{11}$$

$$10) f(x_{10}, \underline{f(x_{10}, z_{10})}) = z_{10}$$

$$\sigma = \{x_{10} \leftarrow y_{11}, z_{10} \leftarrow f(x_{11}, y_{11})\}$$

The paramodulant is:

$$f(y_{11}, x_{11}) = f(x_{11}, y_{11})$$

We obtain (after renaming the variables):

$$12) f(y_{12}, x_{12}) = f(x_{12}, y_{12})$$

$$\text{Resolution (12,1)-(8,1)} \quad \sigma = \{x_{12} \leftarrow b, y_{12} \leftarrow a\}$$

13)  $\square$ .

$\square$

## 9.2. Constraints

The notion of constraint is very natural and appears in many different situations.

Compared with the other notions that have been studied, it is best related to those of logical consequence and subsumption. For example, a theorem that holds for groups is more general (i.e. is valid for more objects) than a theorem that holds for Abelian groups that have the additional property of being commutative. If a theorem holds with less *constraints* on its hypotheses, then obviously (*monotony*, see exercise 3.13 c), it also holds with more of them (subsumption)<sup>4</sup>.

The premises of the theorem can be viewed as constraints and the conclusion as the solution of the constraints (the conclusion holds under the *constraints* of the premises). Every model of the premises (and maybe more) is specified by the formula in the conclusion (equivalent to the logical consequence).

You have already been confronted many times with this notion in a concrete way, for example, in the problem of coloring a graph with the constraint that this must be done with three colors and that two nodes connected by an edge must not have the same color (see example 9.19).

A constraint is a condition to satisfy. The domains (i.e. the sets of possible values for the variables) give their names to the types of constraints. The most studied ones are arithmetic constraints, Boolean constraints, constraints on strings, constraints on trees ((dis-) equations on terms), and constraints on finite domains.

The main problems that arise are:

- satisfiability (*does there exist a solution?*);
- equivalence/implication between constraints;

---

<sup>4</sup> Kolmogorov's probability theory is also *monotonic*, in the sense that if  $B$  is a consequence of  $A$ , then  $prob(A) \leq prob(B)$ .

- simplification;
- optimization.

DEFINITION 9.3.– A wff  $\mathcal{F}$  of FOL of the form:

$$\mathcal{F}: \bigwedge_{1 \leq i \leq n} R_i^{i_n}(x_{i_1}, \dots, x_{i_n})$$

(or a set  $\{R_i^{i_n}(x_{i_1}, \dots, x_{i_n}) \mid 1 \leq i \leq n\}$ )

with  $R_i^{i_n}$ :  $i_n$ -ary predicate symbol

$\mathcal{F}$  is called a constraint.

$$\text{Var}(\mathcal{F}) = \mathcal{V}$$

Given a domain (set)  $\mathcal{D}$ , the problem of finding a substitution:

$$\sigma : \mathcal{V} \longrightarrow \mathcal{D}$$

enabling us to evaluate  $\mathcal{F}$  to **true** is called the problem of constraint solving.

The projection of a constraint  $C$  on a set of variables  $\mathcal{V} \subseteq \text{Var}(C)$  is a constraint  $C_P$  with  $\text{Var}(C_P) = \mathcal{V}$ , such that:

- if  $\sigma$  is a solution of  $C$ , then  $\sigma$  is a solution of  $C_P$ ;
- if  $\text{dom}(\sigma_R) = \mathcal{V}$  and  $\sigma_R$  solution of  $C_P$ , then there exists an extension of  $\sigma_R$  to  $\text{Var}(C)$  that is a solution of  $C$ .

The definition of a projection simply means: “we only deal with the subset of constraints containing some given variables of the problem”.

The procedure PLC specifies the abstract interpreter that permits us to handle clauses with constraints. We give a few examples on using constraints in a logic programming language (Prolog 3).

EXAMPLE 9.14.– (logical connectives). Assuming that we do not have Boolean constraints (although we do), we want to define the usual logical connectives (see section 3.1) as arithmetic operations, using arithmetic constraints, and encoding **T** by 1 and **F** by 0:

$$\text{not}(X,Y) \rightarrow \{ Y=1-X, 0 \leq X \leq 1, 0 \leq Y \leq 1 \};$$

$$\text{and}(X,Y,Z) \rightarrow \{ Z=X \times Y, 0 \leq X \leq 1, 0 \leq Y \leq 1 \};$$

```

procedure PLC;
input:
  A Prolog list:  $\langle C_1, \dots, C_r \rangle$  of non-negative Horn clauses
  (with or without constraints):
  % Possibly Prog:  $\langle \rangle$ 
   $C_i : A_i \rightarrow B_1^i, \dots, B_{n_i}^i \parallel \{R_i\}$  ( $n \geq 0; 1 \leq i \leq r$ )
  A negative clause (with or without constraints) (the question):
  Ques:  $P_1, \dots, P_s \parallel \{Q\}$  ( $s \geq 1$ )
  % Possibly Ques:  $\{Q\}$ 
output: (if the program halts)
   $\square$  or no
begin
  Res  $\leftarrow \langle P_1, \dots, P_s \rangle$ ;
  X  $\leftarrow P_1$ ;
  while Res  $\neq \square$  and Res  $\neq no$ 
  do
  if exists j (take the smallest one among the untreated clauses)
  such that  $(A_j \doteq X) \cup R_j \cup Q$  has a solution  $\sigma$ 
  [We memorize the other possible choices that we will return to
  in case of failure AND of success]
  then Res  $\leftarrow \sigma \langle B_1^j, \dots, B_{n_j}^j \rangle \circ \langle Res - \langle X \rangle \rangle$ 
  %  $\circ$  denotes list concatenation
  % Application of resolution with some strategy
  % We handle a stack: last in first out
  X  $\leftarrow$  first literal in Res
  else Res = no
  enddo
end
  
```

Figure 9.3. Abstract interpreter for PL language with constraints

$or(X,Y,Z) \rightarrow \{ Z=(X+Y)-X \times Y, 0 \leq X \leq 1, 0 \leq Y \leq 1 \};$

$implies(X,Y,Z) \rightarrow \{ Z=1-(X \times (1-Y)), 0 \leq X \leq 1, 0 \leq Y \leq 1 \};$

$equiv(X,Y,Z) \rightarrow \{ Z=(1-(X \times (1-Y))) \times (1-(Y \times (1-X))) ,$

$0 \leq X \leq 1, 0 \leq Y \leq 1 \};$

Of course, *implies* could have been written:

$\text{implies}(X,Y,Z) \rightarrow \{ Z=1 - X + X \times Y, 0 \leq X \leq 1, 0 \leq Y \leq 1 \}$ ; by either transforming the constraint or directly using the equivalence  $(X \Rightarrow Y) \Leftrightarrow (\neg X \vee Y)$ .  $\square$

EXAMPLE 9.15.– (pigeons and rabbits). An example that is frequently given to illustrate constraints is the following.

Determine the  $X$  number of pigeons and the  $Y$  number of rabbits such that together, they consist of 12 heads and 34 legs.

There is no need for a program, we ask the question:

$$\{X + Y = 12, \quad 2 \times X + 4 \times Y = 34\};$$

the answer is:

$$\{X = 7, \quad Y = 5\}. \quad \square$$

EXAMPLE 9.16.– (at the restaurant). This program is probably the most famous program that can be found among the examples of Prolog programs in the literature.

A person who must not eat more than a given number of calories during meals wants to come up with the different menus that consist of, say, less than 10 calories. A program that enables us to get all the menus is the following:

```
light-meal(e,p,d) → starter(e,i) main-course(p,j)
desert(d,k) {0<=i, 0<=j, 0<=k, i+j+k <= 10 };
main-course(p,i) → meat(p,i);
main-course(p,i) → fish(p,i);
starter(radish,1) →;
starter(pasta,6) →;
meat(veal,5) →;
meat(pork,7) →;
fish(sea-bass,2) →;
fish(tuna,4) →;
desert(fruit,2) →;
desert(ice-cream,6) →;
```

By asking the question:

```
light-meal(e,p,d);
```

The person will know what menus are possible with 10 or less calories per meal.  $\square$

EXAMPLE 9.17.– (magic squares). We want to arrange the numbers 1 to 9 in a square in such a way that each row, column, and the two diagonals, all sum to the same constant.

We note:

X1	X2	X3
X4	X5	X6
X7	X8	X9

The program is empty and we ask the question:

enum(X)

(enum(X) enables us to enumerate the integer values of X that satisfy the constraint):

```
enum(X1) enum(X2) enum(X3) enum(X4) enum(X5) enum(X6) enum(X7)
enum(X8) enum(X9)
{ X1#X2, X1#X3, X1#X4, X1#X5, X1#X6, X1#X7, X1#X8, X1#X9,
X2#X3, X2#X4, X2#X5, X2#X6, X2#X7, X2#X8, X2#X9,
X3#X4, X3#X5, X3#X6, X3#X7, X3#X8, X3#X9,
X4#X5, X4#X6, X4#X7, X4#X8, X4#X9,
X5#X6, X5#X7, X5#X8, X5#X9,
X6#X7, X6#X8, X6#X9,
X7#X8, X7#X9,
X8#X9,
1<=X1<=9, 1<=X2<=9, 1<=X3<=9, 1<=X4<=9, 1<=X5<=9, 1<=X6<=9, 1<=X7<=9,
1<=X8<=9, 1<=X9<=9, X1+X2+X3=X4+X5+X6, X1+X2+X3=X7+X8+X9,
X1+X2+X3=X1+X4+X7,
X1+X4+X7=X2+X5+X8, X1+X4+X7=X3+X6+X9, X1+X5+X9=X1+X2+X3,
X3+X5+X7=X1+X2+X3};

{X1 = 8, X2 = 3, X3 = 4, X4 = 1, X5 = 5, X6 = 9, X7 = 6, X8 = 7,
X9 = 2}
{X1 = 8, X2 = 1, X3 = 6, X4 = 3, X5 = 5, X6 = 7, X7 = 4, X8 = 9,
X9 = 2}
{X1 = 6, X2 = 7, X3 = 2, X4 = 1, X5 = 5, X6 = 9, X7 = 8, X8 = 3,
X9 = 4}
{X1 = 6, X2 = 1, X3 = 8, X4 = 7, X5 = 5, X6 = 3, X7 = 2, X8 = 9,
X9 = 4}
{X1 = 4, X2 = 9, X3 = 2, X4 = 3, X5 = 5, X6 = 7, X7 = 8, X8 = 1,
X9 = 6}
{X1 = 4, X2 = 3, X3 = 8, X4 = 9, X5 = 5, X6 = 1, X7 = 2, X8 = 7,
X9 = 6}
```

{X1 = 2, X2 = 9, X3 = 4, X4 = 7, X5 = 5, X6 = 3, X7 = 6, X8 = 1,  
X9 = 8}  
{X1 = 2, X2 = 7, X3 = 6, X4 = 9, X5 = 5, X6 = 1, X7 = 4, X8 = 3,  
X9 = 8}

□

EXAMPLE 9.18.— (the  $n$ -queens problem). The well-known  $n$ -queens problem consists in placing  $n$  queens on an  $n \times n$  chessboard in such a way that no two queens attack each other. This problem is trivial to program. See also exercise 9.4.

We choose  $n=4$ .

The program contains no clause.

In the question,  $L_i$  and  $C_i$  denote the line and column in which queen  $i$  is placed ( $1 \leq i \leq n$ ). For example, the first solution corresponds to:

	R2		
			R4
R3			
		R1	

`% enumerate the solutions`

```
enum(L1) enum(C1) enum(L2) enum(C2) enum(L3) enum(C3) enum(L4)
enum(C4)
```

`% CONSTRAINTS: we begin by the size of the chessboard:`

```
{1<= L1 <=4, 1<= L2 <=4, 1<= L3 <=4, 1<= L4 <=4, 1<= C1 <=4,
1<= C2 <=4, 1<= C3 <=4, 1<= C4 <=4,
```

`% not on the same line or the same column:`

```
L1#L2, L1#L3, L1#L4, L2#L3, L2#L4, L3#L4,
C1#C2, C1#C3, C1#C4, C2#C3, C2#C4, C3#C4,
```

`% not on the same \ diagonals:`

```
L1$-C1#L2$-C2, L1$-C1#L3$-C3,
L1$-C1#L4$-C4, L2$-C2#L3$-C3, L2$-C2#L4$-C4, L3$-C3#L4$-C4,
```

`% not on the same / diagonals:`



```
L1+C1#L2+C2,
L1+C1#L3+C3, L1+C1#L4+C4, L2+C2#L3+C3, L2+C2#L4+C4, L3+C3#L4+C4};
```

```
% THE SOLUTIONS:
```

```
{L1 = 4, C1 = 3, L2 = 1, C2 = 2, L3 = 3, C3 = 1, L4 = 2, C4 = 4}
{L1 = 4, C1 = 3, L2 = 1, C2 = 2, L3 = 2, C3 = 4, L4 = 3, C4 = 1}
{L1 = 4, C1 = 2, L2 = 3, C2 = 4, L3 = 2, C3 = 1, L4 = 1, C4 = 3}
{L1 = 4, C1 = 2, L2 = 3, C2 = 4, L3 = 1, C3 = 3, L4 = 2, C4 = 1}
{L1 = 4, C1 = 2, L2 = 2, C2 = 1, L3 = 3, C3 = 4, L4 = 1, C4 = 3}
{L1 = 4, C1 = 2, L2 = 2, C2 = 1, L3 = 1, C3 = 3, L4 = 3, C4 = 4}
{L1 = 4, C1 = 2, L2 = 1, C2 = 3, L3 = 3, C3 = 4, L4 = 2, C4 = 1}
{L1 = 4, C1 = 2, L2 = 1, C2 = 3, L3 = 2, C3 = 1, L4 = 3, C4 = 4}
{L1 = 3, C1 = 4, L2 = 4, C2 = 2, L3 = 2, C3 = 1, L4 = 1, C4 = 3}
{L1 = 3, C1 = 4, L2 = 4, C2 = 2, L3 = 1, C3 = 3, L4 = 2, C4 = 1}
{L1 = 3, C1 = 4, L2 = 2, C2 = 1, L3 = 4, C3 = 2, L4 = 1, C4 = 3}
{L1 = 3, C1 = 4, L2 = 2, C2 = 1, L3 = 1, C3 = 3, L4 = 4, C4 = 2}
{L1 = 3, C1 = 4, L2 = 1, C2 = 3, L3 = 4, C3 = 2, L4 = 2, C4 = 1}
{L1 = 3, C1 = 4, L2 = 1, C2 = 3, L3 = 2, C3 = 1, L4 = 4, C4 = 2}
{L1 = 3, C1 = 1, L2 = 4, C2 = 3, L3 = 1, C3 = 2, L4 = 2, C4 = 4}
{L1 = 3, C1 = 1, L2 = 2, C2 = 4, L3 = 4, C3 = 3, L4 = 1, C4 = 2}
{L1 = 3, C1 = 1, L2 = 2, C2 = 4, L3 = 1, C3 = 2, L4 = 4, C4 = 3}
{L1 = 3, C1 = 1, L2 = 1, C2 = 2, L3 = 4, C3 = 3, L4 = 2, C4 = 4}
{L1 = 3, C1 = 1, L2 = 1, C2 = 2, L3 = 2, C3 = 4, L4 = 4, C4 = 3}
{L1 = 2, C1 = 4, L2 = 4, C2 = 3, L3 = 3, C3 = 1, L4 = 1, C4 = 2}
{L1 = 2, C1 = 4, L2 = 4, C2 = 3, L3 = 1, C3 = 2, L4 = 3, C4 = 1}
{L1 = 2, C1 = 4, L2 = 3, C2 = 1, L3 = 4, C3 = 3, L4 = 1, C4 = 2}
{L1 = 2, C1 = 4, L2 = 3, C2 = 1, L3 = 1, C3 = 2, L4 = 4, C4 = 3}
{L1 = 2, C1 = 4, L2 = 1, C2 = 2, L3 = 4, C3 = 3, L4 = 3, C4 = 1}
{L1 = 2, C1 = 4, L2 = 1, C2 = 2, L3 = 3, C3 = 1, L4 = 4, C4 = 3}
{L1 = 2, C1 = 1, L2 = 4, C2 = 2, L3 = 3, C3 = 4, L4 = 1, C4 = 3}
{L1 = 2, C1 = 1, L2 = 4, C2 = 2, L3 = 1, C3 = 3, L4 = 3, C4 = 4}
{L1 = 2, C1 = 1, L2 = 3, C2 = 4, L3 = 4, C3 = 2, L4 = 1, C4 = 3}
{L1 = 2, C1 = 1, L2 = 3, C2 = 4, L3 = 1, C3 = 3, L4 = 4, C4 = 2}
{L1 = 2, C1 = 1, L2 = 1, C2 = 3, L3 = 4, C3 = 2, L4 = 3, C4 = 4}
{L1 = 2, C1 = 1, L2 = 1, C2 = 3, L3 = 3, C3 = 4, L4 = 4, C4 = 2}
{L1 = 1, C1 = 3, L2 = 4, C2 = 2, L3 = 3, C3 = 4, L4 = 2, C4 = 1}
{L1 = 1, C1 = 3, L2 = 4, C2 = 2, L3 = 2, C3 = 1, L4 = 3, C4 = 4}
{L1 = 1, C1 = 3, L2 = 3, C2 = 4, L3 = 4, C3 = 2, L4 = 2, C4 = 1}
{L1 = 1, C1 = 3, L2 = 3, C2 = 4, L3 = 2, C3 = 1, L4 = 4, C4 = 2}
{L1 = 1, C1 = 3, L2 = 2, C2 = 1, L3 = 4, C3 = 2, L4 = 3, C4 = 4}
```

```

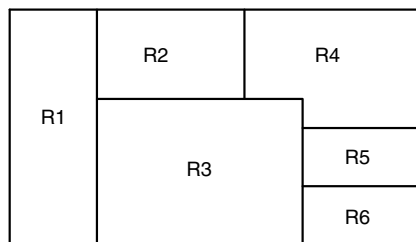
{L1 = 1, C1 = 3, L2 = 2, C2 = 1, L3 = 3, C3 = 4, L4 = 4, C4 = 2}
{L1 = 1, C1 = 2, L2 = 4, C2 = 3, L3 = 3, C3 = 1, L4 = 2, C4 = 4}
{L1 = 1, C1 = 2, L2 = 4, C2 = 3, L3 = 2, C3 = 4, L4 = 3, C4 = 1}
{L1 = 1, C1 = 2, L2 = 3, C2 = 1, L3 = 4, C3 = 3, L4 = 2, C4 = 4}
{L1 = 1, C1 = 2, L2 = 3, C2 = 1, L3 = 2, C3 = 4, L4 = 4, C4 = 3}
{L1 = 1, C1 = 2, L2 = 2, C2 = 4, L3 = 4, C3 = 3, L4 = 3, C4 = 1}
{L1 = 1, C1 = 2, L2 = 2, C2 = 4, L3 = 3, C3 = 1, L4 = 4, C4 = 3}

```

□

EXERCISE 9.4.– Express the  $n$ -queens problem in PL, without using  $\#$ . Compare with example 9.18. □

EXAMPLE 9.19.– (map coloring). The problem consists in coloring all the regions of the map below with three distinct colors, in such a way that no two regions with a common border are colored with the same color.



```

enum(R1) enum(R2) enum(R3) enum(R4) enum(R5) enum(R6)
{1<=R1<=3, 1<=R2<=3, 1<=R3<=3, 1<=R4<=3, 1<=R5<=3, 1<=R6<=3,
R1#R2,R1#R3, R2#R3, R2#R4, R3#R4, R3#R5, R3#R6, R4#R5, R5#R6};
{R1 = 3, R2 = 2, R3 = 1, R4 = 3, R5 = 2, R6 = 3}
{R1 = 3, R2 = 1, R3 = 2, R4 = 3, R5 = 1, R6 = 3}
{R1 = 2, R2 = 3, R3 = 1, R4 = 2, R5 = 3, R6 = 2}
{R1 = 2, R2 = 1, R3 = 3, R4 = 2, R5 = 1, R6 = 2}
{R1 = 1, R2 = 3, R3 = 2, R4 = 1, R5 = 3, R6 = 1}
{R1 = 1, R2 = 2, R3 = 3, R4 = 1, R5 = 2, R6 = 1}

```

□

EXAMPLE 9.20.– We want to compute the minimum  $M$  of an expression  $E$  in a fragment of the plane.

```

minimum(E,M) {1<=X , X<=3 , 0<=Y , 2Y$-$X<=3 , E=X$-$Y};

```

```

> enum(X) enum(Y) enum(E) minimum(E,M) out1(M) {1<=X , X<=3 , 0<=Y ,
2Y$-$X<=3 , E=X$-$Y};

```

```
{X = 3, Y = 3, E = 0, M = 0}
{X = 3, Y = 2, E = 1, M = 1}
{X = 3, Y = 1, E = 2, M = 2}
{X = 3, Y = 0, E = 3, M = 3}
{X = 2, Y = 2, E = 0, M = 0}
{X = 2, Y = 1, E = 1, M = 1}
{X = 2, Y = 0, E = 2, M = 2}
{X = 1, Y = 2, E = -1, M = $-$1}
{X = 1, Y = 1, E = 0, M = 0}
{X = 1, Y = 0, E = 1, M = 1}
$-$1
```

□

EXAMPLE 9.21.- (non-linear constraints?). Sometimes, they can be handled. The Pythagorean theorem is one such example.

```
enum(Z) enum(Y) enum(X) {Z*Z=X*X + Y*Y, 0<=X<=10, 0<=Y<=10, 0<=Z<=200}
{Z = 0, Y = 0, X = 0}
{Z = 1, Y = 0, X = 1}
{Z = 1, Y = 1, X = 0}
{Z = 2, Y = 0, X = 2}
{Z = 2, Y = 2, X = 0}
{Z = 3, Y = 0, X = 3}
{Z = 3, Y = 3, X = 0}
{Z = 4, Y = 0, X = 4}
{Z = 4, Y = 4, X = 0}
{Z = 5, Y = 0, X = 5}
{Z = 5, Y = 3, X = 4} (*)
{Z = 5, Y = 4, X = 3} (*)
{Z = 5, Y = 5, X = 0}
{Z = 6, Y = 0, X = 6}
{Z = 6, Y = 6, X = 0}
{Z = 7, Y = 0, X = 7}
{Z = 7, Y = 7, X = 0}
{Z = 8, Y = 0, X = 8}
{Z = 8, Y = 8, X = 0}
{Z = 9, Y = 0, X = 9}
{Z = 9, Y = 9, X = 0}
{Z = 10, Y = 0, X = 10}
{Z = 10, Y = 6, X = 8} (*)
{Z = 10, Y = 8, X = 6} (*)
{Z = 10, Y = 10, X = 0}
```

In the constraints, we allowed triangles with collinear sides. To eliminate them, it would suffice to add  $1 \leq X$ ,  $1 \leq Y$  in the constraint to obtain the non-degenerate triangles (those marked with (\*)) as solutions. □

EXAMPLE 9.22.– (syntactic analysis). Grammar rules:

$$S \rightarrow c$$

$$S \rightarrow aSb$$

$$S \rightarrow eSd$$

The program:

$$\text{Word}(u) \rightarrow \{u="c"\} ;$$

$$\text{Word}(u) \rightarrow \text{Word}(v) \{u="a".v."b"\} ;$$

$$\text{Word}(u) \rightarrow \text{Word}(v) \{u="e".v."d"\} ;$$

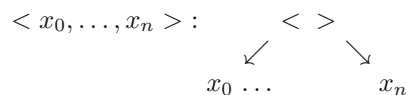
□

EXAMPLE 9.23.– (scalar product).

```
scalprod(<>, <>, 0) ->;
scalprod(<a>.X, <b>.Y, u+a*b) -> scalprod(X,Y,u);
```

```
scalprod(<2,2>,X,14) scalprod(X,<1,5>,23);
{X = <3,4>}
```

where  $\langle \rangle$  denotes a tuple, i.e. a tree whose root is labelled by  $\langle \rangle$ .



□

EXAMPLE 9.24.– (palindrome on strings).  $\text{pal}(\langle \rangle) \rightarrow$ ;

$$\text{pal}(v) \rightarrow \text{pal}(u) \{v=\langle a \rangle.u.\langle a \rangle\};$$

□

EXAMPLE 9.25.– (reverse on strings). List  $y$  is the reverse of list  $x$  if their concatenation is a palindrome:

(uses the following property: the length of the list resulting from the concatenation of a list and its reverse is always an even number)

$$\text{reverse}(x,y) \rightarrow \text{pal}(u) \{u=x.y, x::m, y::n, m=n\};$$

$x::m$  means: “the length of string  $x$  is  $m$ ”.

□

EXAMPLE 9.26.– (in a database). (Example given by K. Marriott and J. Stuckey)

A firm wants to reward the faithful employees (defined as those who have stayed at least 10 years in the firm). The part of the database involved with this problem is the following:

```
works(T,empl3,work1,dept1) → {1985 ≤ T ≤ 1987};
works(T,empl3,work2,dept1) → {T = 1988};
works(T,empl3,work3,dept3) → {1989 ≤ T ≤ 1996};
works(T,empl4,work1,dept1) → {1994 ≤ T ≤ 1995};
works(T,empl2,work1,dept1) → {1988 ≤ T ≤ 1991};
works(T,empl2,work2,dept3) → {1992 ≤ T ≤ 1995};
works(T,empl2,work4,dept4) → {T = 1996};
works(T,empl1,work4,dept4) → {1980 ≤ T ≤ 1996};
faithful-employee(X) → works(T1,X,Z,U) works(T2,X,V,W)
{10 ≤ T2 - T1};
```

*Nb:* in general, when there are variables we are not interested in (such as Z, U, V, W in the question), we have the possibility of replacing them by  $\_$  so as not to choose names for them.

Question: `faithful-employee(X)` ;

Answer:  $\{X = empl3\}$      $\{X = empl1\}$ . □

REMARK 9.7.– The substitution  $\{ N=0, M=ss(K), K=ss(K) \}$  in the answer to question 2 in the table, corresponds to the generation of an infinite, rational tree (no test for cycles in the unification algorithm, see exercise 4.1). □

### 9.3. Second Order Logic (SOL): a few notions

During the 19th Century and up to the 1910s, in the formal systems used by logicians–mathematicians (Frege, Zermelo, etc.), the usage of higher-order quantifiers (for example, on sets) was common. Formulas and proofs of infinite lengths were also permitted.

It is only during the 1920s (in particular, thanks to Skolem and Hilbert) that FOL started being particularly important.

Using higher-order seems natural because we may wonder “why should we limit the expressive power of the logic we are using?”, i.e. the nature of the objects that can be quantified.

Programming with and without constraints: some comparative examples	With the same declarative semantics
Without constraints	With constraints
$sum(0, N, N) \rightarrow;$	$sum(0, N, N) \rightarrow;$
$sum(ss(N), M, ss(K)) \rightarrow sum(N, M, K) ;$	$sum(ss(N), M, ss(K)) \rightarrow \{N+M=K\}$
<b>question 1:</b>	<b>question 1:</b>
$sum(N, M, K) ;$	$sum(N, M, K) ;$
$\{N=0, K=M\}$	$\{N=0, K=M\}$
$\{N=ss(0), K=ss(M)\}$	$\{N=ss(N'), M=M', K=ss(M'+N')\}$
$\{N=ss(ss(0)), K=ss(ss(M))\}$	
...	
<b>question 2:</b>	<b>question 2:</b>
$sum(N, M, K) sum(N, M, ss(K)) ;$	$sum(N, M, K) sum(N, M, ss(K)) ;$
$\{N=0, M=ss(K), K=ss(K)\}$	$\{N=0, M=ss(K), K=ss(K)\}$
$\{N=ss(0), M=ss(K'), K=ss(ss(K')), K'=ss(K')\}$	
...	
<b>question 3:</b>	<b>question 3:</b>
$eq(N, ss(ss(0))) sum(N, M, K) sum(N, M, ss(K)) ;$	$eq(N, ss(ss(0))) sum(N, M, K) sum(N, M, ss(K)) ;$
$\{N=ss(ss(0)), M=ss(K'), K=ss(ss(ss(K'))), K'=ss(K')\}$	fail

The answer is clear when we take into account the *properties* of the logic under consideration, other than its expressive power, e.g. existence of denumerable models, semi-decidability, compactness, and completeness, which are useful, in particular, for its automation.

FOL enables us to express many things, in particular, in mathematics, and it has “nice” properties, but some common concepts such as infinite sets, finite sets, well-ordered sets, continuous functions, etc. cannot be expressed in FOL.

SOL (and more generally so-called higher-order logics) enables us to express these concepts and has (have) another characteristic which is that it (they) can *greatly* reduce the length of proofs of a lower order (for infinitely many formulas).

The notion of expressivity or expressive power corresponds to the models that the formulas of the logic under consideration *a* to characterize in an *exclusive* way (meaning some models and they alone).

Characteristic: explicit quantification on predicates and functions.

Advantage: greater expressive power.

Disadvantage: we lose some of the interesting properties of FOL: Löwenheim–Skolem, compactness, and completeness.

We give a few examples that show its usefulness.

EXAMPLE 9.27.– Leibniz’s law (see section 9.1). □

EXAMPLE 9.28.– (see also example 9.28).

The induction axiom, *intensional version*:

$$\forall P((P(0) \wedge \forall x(P(x) \Rightarrow P(\text{succ}(x))) \Rightarrow \forall yP(y))$$

The induction axiom, *extensional version*:

$$\forall S. \text{ if } [ S \subseteq \mathbb{N} \text{ and } 0 \in S \text{ and (if } n \in S \text{ then } s(n) \in S) ] \text{ then } S = \mathbb{N}. \quad \square$$

EXAMPLE 9.29.– *S* is a *well-ordered set* (meaning that every non-empty subset has a least element):

$$\forall S \exists x(x \in S \Rightarrow \exists y(y \in S \wedge \forall z(z \in S \Rightarrow y \leq z)))$$

This property is expressed in *weak SOL with successor*, which is *decidable* with a non-elementary complexity, meaning that for any decision algorithm and integer *k*, there exist formulas of length *n* such that the decision requires

$$\left. \begin{matrix} \vdots \\ 2^n \\ 2 \end{matrix} \right\} k \text{ units of time.} \quad \square$$

EXAMPLE 9.30.– (torsion group). This example is often used to show the limits of the expressive power of FOL.

An Abelian group  $G$  is a *torsion group* iff every  $a \in G$  has a finite order, i.e.  $\exists n \geq 1. a^n = e^G$  ( $e^G$  denotes the identity in  $G$ ).

It can also be defined as follows:

$\forall X \in \mathcal{P}(G). x \in X \wedge [\forall y \in X. (x \circ y) \in X] \Rightarrow e^G \in X$  ( $\mathcal{P}(G)$  denotes the set of subsets of  $G$ .)

Or, using an infinitary logic (i.e. that allows formulas of infinite length):

$$\forall x[x = e^G \vee x^2 = e^G \vee x^3 = e^G \vee \dots]. \quad \square$$

EXAMPLE 9.31.– (finite set). We first show that the property of being finite (finiteness) *cannot* be expressed by a set of wffs of FOL whose models would *exclusively* be finite sets (obviously of any arbitrary cardinality).

To prove this impossibility, we prove that if such a set of wffs of FOL existed, then it would also admit *infinite* models.

Assume that  $\Sigma$ , a set of wffs of FOL, has finite models of arbitrary cardinalities.

Consider the formulas of FOL that specify that there exist at least  $n$  (distinct) elements (see also theorem 9.1):

$$\varphi_n: \exists x_1, \dots, \exists x_n(x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \dots \wedge x_{n-1} \neq x_n)$$

where the conjunction contains  $n(n-1)/2$  disequations.

We define the set  $\phi = \Sigma \cup \{\varphi_n \mid n \in \mathbb{N}\}$ .

It is clear that the set  $\phi_p = \Sigma \cup \{\varphi_n \mid n \leq p, p \in \mathbb{N}\}$  is a subset of  $\phi$  and is satisfiable (as  $\Sigma$  admits models of all cardinalities and  $\phi_p$  has a model of cardinality at least  $p$ ).

This property holds for all  $p$ , hence, as every finite subset of  $\phi$  is satisfiable, by the compactness theorem for FOL (see theorem 5.8), the infinite set  $\phi$  also admits a model. This model is *infinite* (See also the proof of theorem 9.1.).

However, a model of a set of formulas is a model of *all* its formulas; hence, the infinite model of  $\phi$  is an infinite model of  $\Sigma$ . Therefore,  $\Sigma$  cannot *exclusively* characterize finite sets.  $\square$



However, it is possible to express finiteness in SOL. According to Dedekind's definition, *a set is finite iff there is no bijection from this set onto one of its proper subsets*. In other words,  $E$  is finite iff every injective function  $E \rightarrow E$  is onto (this is not the case, e.g. for  $\mathbb{N}$ , with  $f: \mathbb{N} \rightarrow \mathbb{N}$  and  $f(x) \mapsto 2 \times x$ ).

We propose three translations.

The first translation:

$$\forall f(\forall x\forall y((f(x) = f(y)) \Rightarrow (x = y)) \Rightarrow \forall x\exists y(x = f(y)))$$

The second translation (there are only quantifications on predicate variables):

$$\forall F( \underbrace{(\forall x\exists!yF(x, y))}_{F \text{ is a function}} \wedge \underbrace{\forall x\forall y\forall z((F(x, z) \wedge F(y, z)) \Rightarrow (x = y))}_{F \text{ injective}} \Rightarrow \underbrace{\forall x\exists yF(y, x)}_{F \text{ onto}} )$$

with:

$$\forall x\exists!yF(x, y): \forall x\exists y(F(x, y) \wedge (\exists zF(x, z) \Rightarrow (y = x)))$$

The third translation, “the set whose members have property  $P$  is finite” (see also exercise (5.1 n)):

$$\neg \exists f \left[ \underbrace{\forall x\forall y(f(x) = f(y) \Rightarrow x = y)}_{f \text{ injective}} \wedge \underbrace{\forall x(P(x) \Rightarrow P(f(x)))}_{\text{domain} = \text{codomain}} \wedge \underbrace{\exists y(P(y) \wedge \forall x(P(x) \Rightarrow f(x) \neq y))}_{f \text{ not onto}} \right] \quad \square$$

EXAMPLE 9.32.– (infinite set). Dedekind's definition is sometimes presented for infinite sets.

A set  $E$  is infinite iff there exists an injective function with domain  $E$ , whose codomain is a proper subset of  $E$ :

$$\exists f \left[ \underbrace{\exists z\forall u.z \neq f(u)}_{\text{codomain } f \subsetneq \text{domain } f} \wedge \underbrace{\forall x\forall y((x \neq y) \Rightarrow f(x) \neq f(y))}_{f \text{ injective}} \right]. \quad \square$$

EXAMPLE 9.33.– (infinite set-2). A set is infinite iff it is the domain of a total, transitive, and irreflexive relation:

$$\varphi_\infty : \exists X[\forall u\forall v\forall w(X(u, v) \wedge X(v, w) \Rightarrow X(u, w)) \wedge \forall u\neg X(u, u) \wedge \forall u\exists vX(u, v)] \quad \square$$

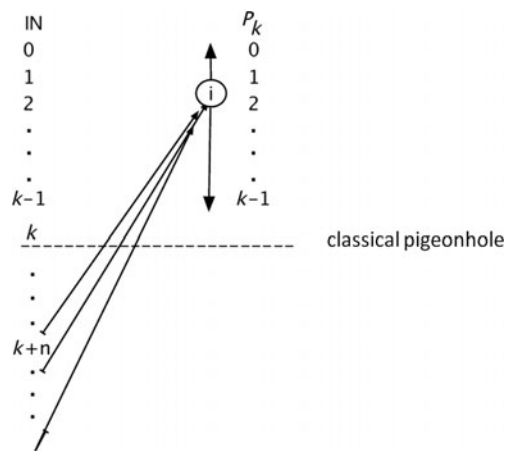
EXAMPLE 9.34.– Sometimes, the pigeonhole principle is formalized by the formula:

$$\forall k \forall f. \mathbb{N} \rightarrow \mathcal{P}_k \exists A \in [\mathcal{P}(\mathbb{N})]_\infty \exists i < k \forall j \in A. f(j) = i$$

where  $[\mathcal{P}(\mathbb{N})]_\infty$  denotes the infinite subsets of  $\mathbb{N}$

and  $\mathcal{P}_k$  the following subset of  $\mathbb{N}$ :  $\{0, 1, \dots, k - 1\}$

Graphically, this principle can be represented by:



□

EXAMPLE 9.35.– In natural language, *Victor Hugo had all the qualities of a great writer*:

$$\forall X (\forall y (G(y) \Rightarrow X(y)) \Rightarrow X(hugo))$$

$X(y)$ :  $y$  has quality  $X$ .

$G(y)$ :  $y$  is a great writer.

□

### 9.3.1. Syntax and semantics

For the sake of readability, we only present what needs to be added to FOL.

#### 9.3.1.1. Vocabulary

- Predicate variables  $\{ X_1^{n_1}, X_2^{n_2}, X_3^{n_3}, \dots \}$ .
- Function variables  $\{ F_1^{n_1}, F_2^{n_2}, F_3^{n_3}, \dots \}$ .

(in FOL, we only have variables denoting individuals)

DIGRESSION 9.1.– (variables 3)<sup>5</sup>. In this context too, variables can be replaced by other symbols (syntactic objects) and do not denote quantities that vary (as in mathematics and physics).  $\square$

### 9.3.1.2. Syntax

- Atomic formulas can be constructed using predicate and function variables.
- If  $\varphi$  is a wff, then  $\forall X_i^{n_i} \varphi, \exists X_i^{n_i} \varphi, \forall F_i^{n_i} \varphi, \exists F_i^{n_i} \varphi$  are also wffs.

### 9.3.1.3. Semantics

Consider a structure  $\mathcal{M} = \langle D, \mathcal{F}, \mathcal{R} \rangle$  and an interpretation  $\mathcal{I}$ :

- the interpretation assigns to  $F_i^{n_i}$  a function  $D^{n_i} \rightarrow D$  in  $\mathcal{F}$ ;
- the interpretation assigns to  $X_i^{n_i}$  a relation  $R^{n_i} \subset D^{n_i}$  in  $\mathcal{R}$ ;
- $\mathcal{M} \models_{\mathcal{I}} \forall F_i^{n_i} \varphi$  iff for all  $n_i$ -ary functions  $f$  in  $\mathcal{F}$ ,  $\mathcal{M} \models_{\mathcal{I}} \varphi[F_i^{n_i} | f]$ ;
- $\mathcal{M} \models_{\mathcal{I}} \exists F_i^{n_i} \varphi$  iff there exists an  $n_i$ -ary function  $f$  in  $\mathcal{F}$  such that  $\mathcal{M} \models_{\mathcal{I}} \varphi[F_i^{n_i} | f]$ ;
- $\mathcal{M} \models_{\mathcal{I}} \forall X_i^{n_i} \varphi$  iff for all  $n_i$ -ary relations  $R$  in  $\mathcal{R}$ ,  $\mathcal{M} \models_{\mathcal{I}} \varphi[X_i^{n_i} | R]$ ;
- $\mathcal{M} \models_{\mathcal{I}} \exists X_i^{n_i} \varphi$  iff there exists an  $n_i$ -ary relation  $R$  in  $\mathcal{R}$  such that  $\mathcal{M} \models_{\mathcal{I}} \varphi[X_i^{n_i} | R]$ .  $\square$

The compactness theorem (see theorem 5.8) no longer holds in SOL:

THEOREM 9.1.– (non-compactness of SOL). *There exists an unsatisfiable set of wffs in SOL only containing finite subsets that are satisfiable.*

PROOF.– Given  $n \in \mathbb{N} (n \geq 2)$ , consider the formula corresponding to the proposition *there exists at least  $n$  objects*:

$$\varphi_n: \exists x_1, \dots, \exists x_n (x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge \dots \wedge x_{n-1} \neq x_n)$$

where the conjunction contains  $n(n-1)/2$  inequalities.

It is clear from the intended interpretation that each  $\varphi_n$  is satisfiable in models with universes  $D$  of cardinalities at least  $n$  (in particular,  $\varphi_n$  is satisfiable in infinite universes) and unsatisfiable in universes  $D$  with  $\text{card}(D) < n$ .

Consider the set:

$$S: \{\varphi_2, \varphi_3, \dots\}$$

---

<sup>5</sup> See also digressions 3.3 and 5.2.

$S$  is not finitely satisfiable. Indeed, assume  $S$  is finitely satisfiable. Consider a universe  $D_f$  such that  $\text{card}(D_f) = n$ , in which  $S$  is satisfied, i.e. that all the formulas in  $S$  are satisfied. But there are infinitely many formulas in  $S$  that are falsified in  $D_f$ : all the formulas  $\varphi_{n+j}$  for  $j \geq 1$ . *Contradiction.*

We now consider the set  $S_{nc} = \{\neg\varphi_\infty, \varphi_2, \varphi_3, \dots\}$

where  $\varphi_\infty$  is defined in example 9.33,  $S_{nc}$  is clearly unsatisfiable.

Every finite subset of  $S_{nc}$  is satisfiable, because every  $\{\neg\varphi_\infty, \varphi_2, \varphi_3, \dots, \varphi_n\}$

( $n \in \mathbb{N}$ ) and  $\{\varphi_i, \dots, \varphi_j\}$  ( $i, j \in \mathbb{N}$ ) admit finite models (it is not contradictory to admit finite models and not to admit any infinite model).

The set  $S_{nc}$  therefore proves the theorem.  $\square$

EXAMPLE 9.36.– Specify that a graph is three-colorable, i.e. that we can color all the nodes of the graph with three colors in such a way that two nodes connected by an edge are not colored the same way.

$$\exists C_1 \exists C_2 \exists C_3 \forall x [(C_1(x) \vee C_2(x) \vee C_3(x)) \wedge \forall y (A(x, y) \Rightarrow \neg(C_1(x) \wedge C_1(y)) \wedge \neg(C_2(x) \wedge C_2(y)) \wedge \neg(C_3(x) \wedge C_3(y)))]$$

$C_i(x)$  ( $1 \leq i \leq 3$ ): node  $x$  has color  $i$

$A(x, y)$ : there is an edge between nodes  $x$  and  $y$ .  $\square$

EXERCISE 9.5.– (Skolemization and equivalence) (see section 5.5.1.1.).

Prove the equivalence:

$$\forall x \exists y F[x, y] \Leftrightarrow \exists \mathbf{f} \forall x F[x, \mathbf{f}(x)]$$

where  $F[x, y]$  denotes a wff of FOL only containing variables in  $\{x, y\}$  and  $\mathbf{f}$  a function symbol that does not occur in  $F[x, y]$ .  $\square$

EXERCISE 9.6.– Give models of the following formulas:

a)  $\forall P (\exists x P(x) \Rightarrow \exists z (P(z) \wedge \forall y (P(y) \Rightarrow \neg R(y, z))))$

b)  $\forall C ((\exists x C(x) \wedge \forall u \forall y ((C(u) \wedge A(u, y)) \Rightarrow C(y))) \Rightarrow \forall z C(z))$

c)  $\exists C \exists x \exists y (C(x) \wedge \neg C(y) \wedge \forall u \forall v (C(u) \wedge A(u, v) \Rightarrow C(v))$   $\square$

## Chapter 10

# Non-classical Logics

In the following sections, we shall present some of these logics. The study can be partially based on the notions that were introduced in the previous chapters.

As usual, the presentation will contain motivations, historic landmarks, applications, similarities, and differences with what is already known, and finally indications on how to reason with these logics.

A common feature of these logics is their philosophical origin, which can be explained as a return to the roots: analysis of discourse in natural language, of what is true, what is false, what is neither true nor false, what is necessary, what is contingent, etc.

### 10.1. Many-valued logics

One of the first conventions that we adopted in the study of logic was to state (in accordance with mathematics) that a reasoning is correct if and only if *it is impossible* for the premises to be **true** and the conclusion to be **false**. But common sense shows that things are not that simple, when we want to begin the analysis *before* any simplification choice has been made.

Indeed, among the syntactically correct sentences, those we are interested in are declarative sentences and more precisely propositions (i.e. the sets of declarative sentences that are synonyms).

There are declarative sentences that are never assigned truth values, for example,

*Honesty is beautiful*

whose meaning can only be metaphorical. Others such as

*The robot is in the room*

can sometimes be true and sometimes be false. . .

The study of many-valued logics, which we shall also refer to as  $p$ -valued logics or  $n$ -valued logics (implicitly stating that  $p, n \geq 3$ ), originated, similar to many other subjects, with the work of Aristotle, who considered the *future contingents* (see section 10.3), i.e. what may or may not occur. The example considered by Aristotle was the proposition:

There will be a sea battle tomorrow.

which is neither true nor false (today) and suggests a third truth value. Łukasiewicz (following Aristotle) believed that if we do not accept that declarative sentences about the future are neither true nor false, then we have to accept fatalism (philosophical doctrine according to which all events are predetermined by fate).

Some historic landmarks:

- during the Middle Ages, little research on the topic;
- end of the 19th Century and beginning of the 20th Century: they are definitively accepted;
- some important names: MacColl, C.S. Peirce (around 1909), N.A. Vasil'ev, Łukasiewicz (starting in 1920), and Post (starting in 1921);
- three-valued logics have been particularly studied;
- there exist  $\infty$ -valued logics (infinitely-valued logics), in particular, the  $L_{\aleph_1}$  logic (see section 10.2).

The introduction by Łukasiewicz of  $p$ -valued logics was preceded by some profound philosophical considerations.

According to historians and philosophers of logic, two principal origins of  $p$ -valued logics seem to have been:

1) the theory of contradictory objects, that states the existence of objects having contradictory properties. It seems like Łukasiewicz believed that non-contradictory objects did not exist;

2) (for three-valued logic) the notions of (non)determinism, causality, free will, etc.

Łukasiewicz studied the problems of induction and probabilities seriously. In his first papers, the logical values depended on the finite number of considered individuals, for example:

- To  $x^2 = 1$ , we assign value  $2/3$  in the universe  $\{-1, 0, 1\}$
- To  $x^2 = 1$ , we assign value  $2/5$  in the universe  $\{-2, -1, 0, 1, 2\}$

We reduce the algebraic formalism to a minimum for the treatment of  $p$ -valued logics. The adopted formalization is sufficient for our needs and can easily be generalized.

Using logics with three truth values can cause the loss of powerful techniques such as the law of excluded middle. For example, the following reasoning:

$$\frac{P \Rightarrow Q \quad \neg P \Rightarrow R}{Q \vee R}$$

is not a correct reasoning in three-valued logic.

EXAMPLE 10.1.– A finer analysis of programs can be performed with a three-valued logic than with a two-valued one.

For example, if  $p$  and  $q$  are Boolean conditions and  $f$  and  $g$  are functions ( $(p \rightarrow f, q \rightarrow g, h)$  means: if  $p$  then  $f$  else (if  $q$  then  $g$  else  $h$ )):

$p$	$q$	$(p \rightarrow f, q \rightarrow g, h)$ computes
<b>T</b>	<b>T</b>	$f$
<b>T</b>	<b>F</b>	$f$
<b>F</b>	<b>T</b>	$g$
<b>F</b>	<b>F</b>	$h$

It seems obvious that the programs  $(p \rightarrow f, f)$  (meaning if  $p$  then  $f$  else  $f$ ) and  $f$  are equivalent; however, whereas  $f$  always computes  $f$ , we can imagine that, for example,  $p : (1/x) \leq 3$ , and as variable  $x$  is real, it can take the value 0, which yields:

$p$	$(p \rightarrow f, f)$ computes
<b>T</b>	$f$
<b>F</b>	$f$
$\perp$	$\perp$

□

DEFINITION 10.1.– (truth functional (extensional) connective). A binary connective  $\odot$  is truth functional iff the truth value of  $A \odot B$  only depends on the truth values of  $A$  and  $B$  (i.e. a truth table can be provided for  $\odot$ ). A logic is truth functional iff all its connectives are truth functional. Truth functional connectives are also called extensional connectives (see definition 2.3).

DEFINITION 10.2.– (*n*-valued logic). If  $\mathcal{L}$  is a formal language generated by propositional variables and connectives,  $N \subset \mathbb{N}$ ,  $N = 0, 1, \dots, n-1$  is a finite set of truth values and  $D \subseteq N$  is a set of distinguished values, then  $L = \langle \mathcal{L}, N, D \rangle$  is an *n*-valued logic.

We can take an infinite set for  $N$  (see definition 10.3).

REMARK 10.1.– One fundamental characteristic that is shared by classical and many-valued logics is truth functionality.  $\square$

We shall present the most commonly used three-valued logics, that is, those of Łukasiewicz, Kleene, and Bochvar.

*Łukasiewicz's  $L_3$  logic (1920)*

Basic principle: being able to handle contingent futures. To do so, he introduced the intermediate value  $1/2$ .

The connectives are defined with the following tables:

$P$	$\neg P$
0	1
$1/2$	$1/2$
1	0

$\Rightarrow$	0	$1/2$	1
0	1	1	1
$1/2$	$1/2$	1	1
1	0	$1/2$	1

the other usual connectives are defined as follows:

$$\alpha \vee \beta : (\alpha \Rightarrow \beta) \Rightarrow \beta$$

$$\alpha \wedge \beta : \neg(\neg\alpha \vee \neg\beta)$$

$$\alpha \Leftrightarrow \beta : (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

which yields:

$\vee$	0	$1/2$	1
0	0	$1/2$	1
$1/2$	$1/2$	$1/2$	1
1	1	1	1



$\wedge$	0	1/2	1
0	0	0	0
1/2	0	1/2	1/2
1	0	1/2	1

$\Leftrightarrow$	0	1/2	1
0	1	1/2	0
1/2	1/2	1	1/2
1	0	1/2	1

DEFINITION 10.3.– (interpretation, valuation).

– An interpretation (valuation) is an application: set of wffs  $\rightarrow \{0, 1/2, 1\}$  that respects the truth tables above.

– A tautology or valid wff is a wff that takes the value denoted by 1 (or a value in  $D$ ) for every interpretation.

For general  $p$ -valued logics, we must consider  $N$  instead of  $\{0, 1/2, 1\}$  and  $D$  instead of 1.

What changes and what does not:

$P$	$\neg P$	$P \vee \neg P$	$P \wedge \neg P$	$\neg(P \wedge \neg P)$	$P \Leftrightarrow \neg P$	$P \Rightarrow P$
0	1	1	0	1	0	1
1/2	1/2	<b>1/2</b>	1/2	<b>1/2</b>	1	1
1	0	1	0	1	0	1

$P \vee \neg P$ : tautology in CPC (Classical Propositional Calculus), non-tautological in  $L_3$

$\neg(P \wedge \neg P)$ : tautology in CPC, non-tautological in  $L_3$

$P \Leftrightarrow \neg P$ : contradiction in CPC, satisfiable in  $L_3$

$P \Rightarrow P$ : tautology in CPC, tautology in  $L_3$

A formal system for this logic:

$\mathcal{S}_{L_3} = \{\mathcal{L}_3, \mathcal{R}, \mathcal{A}\}$ , with:

$\mathcal{L}_3 = \mathcal{L}_0$  % i.e. it is the same as for  $\mathcal{S}_1$

$\mathcal{R} = \{MP\}$

$\mathcal{A}$ :

$$3L1) \quad A \Rightarrow (B \Rightarrow A)$$

$$3L2) \quad (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$$

$$3L3) \quad (\neg B \Rightarrow \neg A) \Rightarrow (A \Rightarrow B)$$

$$3L4) \quad ((A \Rightarrow \neg A) \Rightarrow A) \Rightarrow A$$

REMARK 10.2.– In  $L_3$ , the deduction theorem (see remark 3.14) *does not hold*.  $\square$

The truth tables could have been established the following way:

$$v(\neg \mathcal{A}) = 1 - v(\mathcal{A})$$

$$v(\mathcal{A} \vee \mathcal{B}) = \max\{v(\mathcal{A}), v(\mathcal{B})\}$$

$$v(\mathcal{A} \wedge \mathcal{B}) = \min\{v(\mathcal{A}), v(\mathcal{B})\}$$

$$v(\mathcal{A} \Rightarrow \mathcal{B}) = \begin{cases} 1 & \text{if } v(\mathcal{A}) \leq v(\mathcal{B}) \\ 1 - v(\mathcal{A}) + v(\mathcal{B}) & \text{if } v(\mathcal{A}) > v(\mathcal{B}) \end{cases}$$

or:

$$v(\mathcal{A} \Rightarrow \mathcal{B}) = \begin{cases} 1 & \text{if } v(\mathcal{A}) \leq v(\mathcal{B}) \\ v(\mathcal{B}) & \text{if } v(\mathcal{A}) > v(\mathcal{B}) \end{cases}$$

Sometimes, the latter is expressed as:

$$v(\mathcal{A} \Rightarrow \mathcal{B}) = \min\{1, 1 - v(\mathcal{A}) + v(\mathcal{B})\}$$

$$v(\mathcal{A} \Leftrightarrow \mathcal{B}) = v((\mathcal{A} \Rightarrow \mathcal{B}) \wedge (\mathcal{B} \Rightarrow \mathcal{A})) = 1 - |v(\mathcal{A}) - v(\mathcal{B})|$$

$$v(\mathcal{A} \Leftrightarrow \mathcal{B}) = \begin{cases} 1 & \text{if } v(\mathcal{A}) = v(\mathcal{B}) \\ < 1 & \text{if } v(\mathcal{A}) \neq v(\mathcal{B}) \end{cases}$$

These last definitions naturally lead to considering  $n$ -valued logics with  $n \geq 4$ , in particular  $\infty$ -valued logics.  $\square$

*Kleene's three-valued logics (1938 and 1952)*

Basic principle: give a value that means undecidable (and not an intermediate value) to the mathematical assertions that are true or false, but cannot be proved or refuted.

REMARK 10.3.– This principle is not respected by the laws of excluded middle and of non-contradiction. Doing otherwise would lead to the non-respect of truth functionality. For example, we would assign **true** to  $p \vee \neg p$ . Assume the value of  $p$  is  $1/2$  and that  $\neg p$  is replaced by  $\neg q$  with the same truth value, then the value of  $p \vee \neg q$  is...  $1/2$ , because  $\neg 1/2 = 1/2$  (table for  $\neg$ ) and  $1/2 \vee 1/2 = 1/2$  (table for  $\vee$ ).  $\square$

In so-called *strong* logic (1938) we assign the value  $i$  when the values 0 (false) and 1 (true) are not sufficient to conclude (same principle as in Łukasiewicz's  $L_3$ ).

$\Rightarrow$	0	$i$	1
0	1	1	1
$i$	$i$	$i$	1
1	0	$i$	1

$\Leftrightarrow$	0	$i$	1
0	1	$i$	0
$i$	$i$	$i$	$i$
1	0	$i$	1

The other truth tables are the same as in  $L_3$ .

In so-called *weak* logic (1952), the occurrence of value  $i$  in a sub-formula forces the formula to also have value  $i$ .

$\Rightarrow$	0	$i$	1
0	1	$i$	1
$i$	$i$	$i$	$i$
1	0	$i$	1

$\vee$	0	$i$	1
0	0	$i$	1
$i$	$i$	$i$	$i$
1	1	$i$	1

$\wedge$	0	$i$	1
0	0	$i$	0
$i$	$i$	$i$	$i$
1	0	$i$	1

*Bochvar's three-valued logic (1938)*

Basic principle: try to eliminate paradoxes, such as

*This sentence is wrong.*

*(It is true if it is false, false if it is true)*

Similarly to Kleene's weak logic, if a sub-formula has value  $i$ , then the formula also has value  $i$ .

For Bochvar, a proposition can be *significant* (it is true or false) or *without signification* (a paradox). He proposed two types of connectives: the *internal*

connectives and the *external* connectives. The connectives we are interested in here are the former, whose truth tables coincide with Kleene's weak connectives.

Paradoxes are not eliminated:

*This sentence is false or undetermined.*

*(It is true if it is false or undetermined, it is false or undetermined if it is true.)*

### 10.1.1. How to reason with $p$ -valued logics?

We show in the following example how to extend the method of semantic tableaux for classical logic (see section 3.2), to handle  $p$ -valued logics ( $p$ : finite). The method is general of course, as will be evidenced by the example.

In classical logic (two-valued), the method of semantic tableaux enumerates all the models (partial models in FOL) of a finite set of formulas  $S$ . When it is not possible to construct any model, we conclude that  $S$  is unsatisfiable (see section 3.2).

#### 10.1.1.1. Semantic tableaux for $p$ -valued logics

We consider  $L = \langle \mathcal{L}, N, D \rangle$  and a set of symbols that do not belong to the vocabulary of  $\mathcal{L}$ :  $a_0, a_1, a_2, \dots, a_{n-1}$ . These  $a_i, 0 \leq i \leq n-1$  will be used to indicate the  $n$  truth values of the (sub-)formulas in the tableau.

We discover the new method by analogy with the analogy for two-valued logic.

two-valued logic:

- 1) We are interested in the interpretations  $v$  such that  $v(X) = 1$  (i.e. **true**).
- 2) Branch  $B$  is *closed* iff  $P \in B$  and  $\neg P \in B$  for  $P$  (in  $X$ ): atomic.
- 3)  $X$  is valid iff tableau closed for  $\neg X$ .

$p$ -valued logics ( $p \geq 3$ ):

- 1) We are interested in the interpretations  $v$  such that  $v(X) \in D$ .
- 2) Branch  $B$  is *closed* iff either  $a_i(P) \in B, a_j(P) \in B$  for  $i \neq j$  and  $P$  atomic or  $B$  contains a formula on which no rule can be applied.
- 3)  $X$  valid iff the tableau is closed for all  $a_j(X)$  with  $j \in (N - D)$ , meaning that  $X$  cannot be evaluated to a value that is not distinguished.

EXAMPLE 10.2.– (N. da Costa). In this example, we show how to adapt the method of semantic tableaux to  $p$ -valued logics. Although this is only an example, it is clear that the method can be applied for any finitely valued logic.

Given the following truth tables for connectives  $\neg$ ,  $\Rightarrow$ ,  $\wedge$ , and  $\vee$  in a three-valued logic, with distinguished values  $D = \{0, 1\}$ .

$P$	$\neg P$
0	2
1	0
2	0

$\Rightarrow$	0	1	2
0	0	0	2
1	0	0	2
2	0	0	0

$\vee$	0	1	2
0	0	0	0
1	0	0	0
2	0	0	2

$\wedge$	0	1	2
0	0	0	2
1	0	0	2
2	2	2	2

In each truth table, the leftmost column corresponds to all possible values of the argument on the left-hand side of the connective (named  $X$  below). The first line corresponds to all possible values for the argument on the right-hand side of the connective (named  $Y$  below).

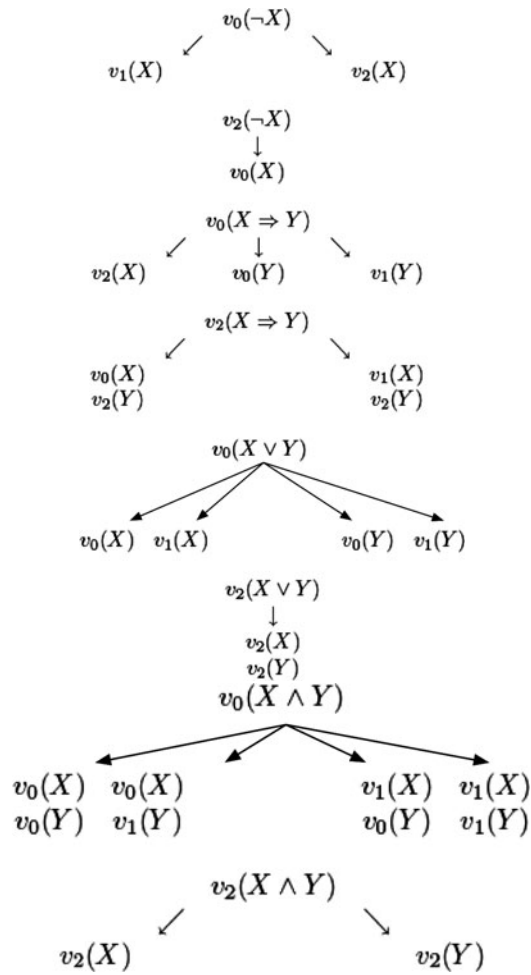
We want to show that the following formula is valid:

$$(P \Rightarrow Q) \Rightarrow (\neg P \vee Q)$$

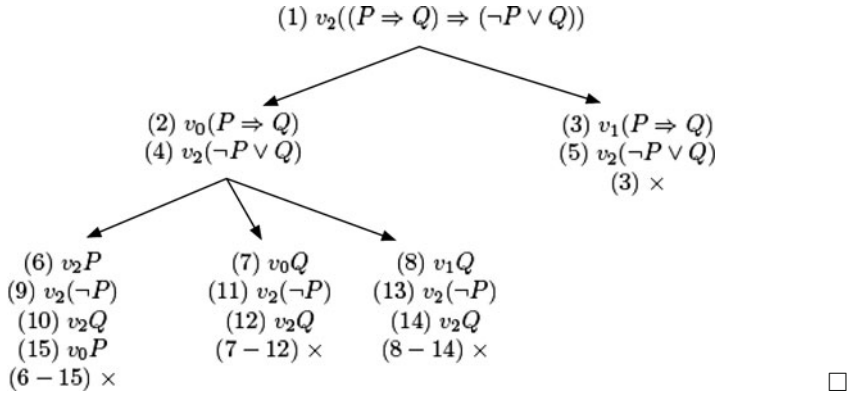
We translate the truth tables (and try to compress their representation).

$v_i(form)$  ( $0 \leq i \leq 2$ );  $form \in \{X, \neg X, X \Rightarrow Y, X \vee Y, X \wedge Y\}$  means  $form$  has value  $i$ .

(Although we do not need the connective  $\wedge$  in this example, we also provide the corresponding rules).



Method: show that it is impossible to evaluate the considered formula to a value that is not distinguished (in this example, value 2).



**10.2. Inaccurate concepts: fuzzy logic**

No man has ever or will ever know anything certain.

*Xenophanes (6th Century BC)*

Similar to the word “model” (see section 5.2), the word “concept” is very frequently used, for example, in AI (and sometimes, it is badly used). It is a notion that has been studied extensively in philosophy.

Among the technical definitions that can be found in the literature, we choose the following one:

Concept: an idea (of a mammal, a triangle, etc.) that is abstract, general, or can be subject to a generalization.

Concept is the rational way of knowing reality.

A distinction must be made between the *concept* and the *object*. The concept is the intensional counterpart<sup>1</sup> of the object<sup>2</sup>.

There are *a priori* concepts, i.e. concepts that are not a result of experience, unity, plurality, etc. and *a posteriori* concepts, i.e. general notions that define classes of objects that are either given or constructed (concept of a mammal, a function, etc). Different philosophical schools claim that only one or the other of these concepts is really a concept.

<sup>1</sup> The *intension* is the set of characters that are (considered as) essential to a class.

<sup>2</sup> What possesses an existence of its own, independently of the knowledge or the idea that cognitive beings can have about it.

Every concept possesses an *extension*, possibly empty. The objects are defined by the concepts that indicate the set of characters that belong (or are supposed to belong) to the objects of the class, and to no other object, for example, *man*: *rational animal*.

When creating concepts, *similarity (analogy)* plays a crucial role. We can also say that the notion of similarity (of which we could say that equality is a particular case<sup>3</sup>) also plays a major role in FL.

We can relate this notion to that of a model. In a model, we choose some characteristics that we wish to be representative and whose goal could be to define the class under consideration.

REMARK 10.4.– The activities of *definition (specification)* and *modeling* are at the core of a computer scientist's activities. They can be viewed from the point of view of the philosophical classification of *constructive definitions* (specification) and *explanatory definitions* (modeling). Example of the former: **append** (see example 6.2) and of the latter **small**: **someone of a size less than (say) 1.70 m**. □

#### *Inaccurate concepts*

A simple way of illustrating the ideas is to oppose mathematical concepts to empirical concepts. The former are *exact* because they do not accept limit or neutral cases (example: *prime number, right triangle,...*). The latter accept neutral cases and are *inaccurate* (example: *big, old, etc.*).

More generally, in natural languages, there are terms that are *vague*, i.e. terms (corresponding to concepts) such that if the extension of the concept is the class denoted by  $\mathcal{C}$ , and if we consider an arbitrary object denoted by  $\mathcal{O}$ , then there is no defined answer to the question: “is  $\mathcal{O}$  an object in class  $\mathcal{C}$ ?”. For example, *very tall, small, bald, almost old, very beautiful, etc.*

The vagueness is different from ambiguity (the same term can denote different objects in different contexts).

Mathematics and logic gave little consideration to vague concepts, more precisely, they use formal languages to avoid these difficulties.

The theory of fuzzy sets and FL (introduced by L. Zadeh in 1965) are an attempt to formalize vague or inaccurate concepts. Fuzzy sets and FL have given rise to many controversies.

---

<sup>3</sup> For example, in practice, saying that two objects have the same color means that the potential differences are not detectable by the sense (or device) that detects colors.



Nowadays, the literature on these topics is huge and we may say that the domain is rather (an example of a vague term!) accepted and respected.

An important distinction: fuzzy  $\neq$  probable

*Fuzziness* is inaccuracy and imprecision. The truth of a fuzzy proposition is a matter of degree. It must be distinguished from the *probable* (i.e. uncertainty as a degree of belief).

The following example clarifies the differences.

EXAMPLE 10.3.– Consider the following propositions:

- a) John is young.
- b) John will be alive next week.

a) is a vague or imprecise proposition (because of the occurrence of the word “young”).

b) is a precise proposition that is true or false, but we do not know that it is. It can be assigned a probability of being true or false, but the probability is not a degree of truth.

From an observational point of view, the vague concept “young” is in principle concerned with only one observation; however, the probability of a young person being alive next week is concerned with several observations of young people.

Most FL are truth functional (see section 10.1), which is not the case for probabilities:

we know that:

- 1)  $prob(p \wedge \neg p) = 0$
- 2)  $prob(\neg p) = 1 - prob(p)$
- 3)  $prob(p \vee q) = prob(p) + prob(q) - prob(p \wedge q)$
- 4)  $prob(p \vee \neg p) = 1$

if for example:

$$(*) prob(p) = 1/2, \text{ hence } prob(\neg p) = 1/2$$

then using (3) with  $q : \neg p$  and replacing  $\neg p$  by  $p$ , which has, according to (\*), the same probability, we obtain:

$$5) prob(p \vee p) = 1/2 + 1/2 - 1/2 = 1/2$$

As a conclusion: in the left-hand side of (4), we replaced an event ( $\neg p$ ) by another event with the same probability ( $p$ ) (see (\*)) and we obtain two different values.

Another example (that also shows the difference with accurate propositions, in particular, the *non-validity of the law of non-contradiction*). From an intuitive point of view, the sentence:

I am old and I am not old.

does not have the value false. This can be formalized, because the basic logic for FL is  $L_{\mathbb{N}_1}$  (i.e. with values in  $\mathbb{R}$ ), which gives  $v(A \wedge \neg A) = \min\{v(A), 1 - v(A)\}$ , and which will in general will be  $\neq 0$ .

Of course,  $\text{prob}(A \wedge \neg A) = 0$ . □

The notion of FL has two meanings: a narrow sense and a wide sense.

For the narrow sense, FL is a logical system whose goal is to formalize approximate reasoning (see below the characterization proposed by L. Zadeh). In this sense, it is an extension of  $p$ -valued logics.

For the wide sense,  $\text{FL} \approx$  theory of fuzzy sets.

A very old paradox seen from a new perspective. A good example of the kinds of problems that are treated by FL is the analysis (of an equivalent version) of the paradox of the heap (see example 8.1).

$X$  denotes the set of all men

$S \subseteq X$  the set of all small men.

We consider a real interval  $P = [0.5, 3]$  that contains all possible sizes.

The function **size**  $h : X \rightarrow P$ .

We assume:

- 1)  $S \neq \emptyset$  (meaning that there exist men who are small).
- 2) No man has a size less than 0.5 m or greater than 3 m (which yields interval  $P$  above).
- 3) There are men of all intermediate sizes – w.r.t. the minimal difference that can be detected by measuring instruments (this will entail no loss of generality in the reasoning).
- 4) If  $x \in S$ ,  $y \in X$ , and  $0 \leq h(y) - h(x) \leq 10^{-3}$ , then  $y \in S$  (we could have chosen a smaller difference in size, e.g.  $10^{-6}$ ).

A man that is 1 mm taller than a small man is also small.

5) If  $x \in S \wedge h(y) \leq h(x)$ , then  $y \in S$ . A man that is smaller than a small man is also small.

We want to prove that:

All men are small.

The proof is simple. Given any man  $y$ , we select a man, say  $x$ , who is without a doubt small. We choose  $x_1$  such that  $h(x_1) - h(x) \leq 10^{-3}$ . By 4,  $x_1 \in S$ .

By iterating (at most  $10^3(h(y) - h(x))$  times), we reach the conclusion that  $y$  is small. (If  $y$  had been smaller than  $x$ , by (5) we would immediately have reached the conclusion).

By applying (4), we prove that all men who are taller than a small man are small.

**Where is the problem?**

In the fact that we did not define (from a mathematical point of view) the set  $S$ , i.e. that we did not give the characteristic function enabling us to tell for any man whether he belongs to the set of small men or not.

The idea to resolve this paradox is to represent the set  $S$  (i.e. the extension of the concept “small man” as a *fuzzy set* (see definition 10.4), i.e. a class of objects for which the transition from membership to non-membership is gradual rather than abrupt.

FL and fuzzy sets were introduced to take what is called *practical reasoning* into account, essentially to model those aspects that are hard to handle with classical logic. An example of practical reasoning is the frequent mix of precise and approximative reasoning that is performed when solving problems.

FL deals with fuzzy propositions such as: *Patricia is extremely intelligent. Most dogs are nice. Michael is much taller than John, and so on.*

One particular characteristic is that not only is the meaning of terms subjective, but it is also local, i.e. restricted to the considered domain of discourse (it is not really the same to be considered as tall by Pygmies or by Scandinavians). Thus, FL can be viewed as a logic whose propositions, connectives, truth values, etc. do not have a

universal value. This implies that inference processes in FL are of a semantic nature rather than a syntactic one.

The creator of FL (L. Zadeh) described it as follows:

Perhaps the simplest way of characterising fuzzy logic is to say that it is a logic of approximate reasoning.

The term *fuzzy logic* is used to describe an imprecise logical system, *FL*, in which the truth-values are fuzzy subsets of the unit interval with linguistic labels such as *true*, *false*, *not true*, *very true*, *quite true*, *not very true* and *not very false*, etc.

Some examples are:

EXAMPLE 10.4.–

Most men are superficial  
Socrates is a man  
-----  
Socrates is *probably* superficial

Michael is *small*  
Michael and John have *approximately* the same height  
-----  
John is *more or less* small

□

The basic idea of fuzzy logic is that predicates (corresponding to properties, relations) denote fuzzy subsets of some universe and the truth values denote fuzzy subsets of the set of values of basic logic, i.e.  $[0, 1]$ .

### Fuzzy sets: definition and fundamental properties

DEFINITION 10.4.– Let  $U$  denote the universe of discourse.

A fuzzy set  $A$  is defined by the generalized characteristic function:

$$\mu_A : U \longrightarrow [0, 1]$$

(the range of a standard characteristic function is  $\{0, 1\}$ , hence ordinary sets are considered as particular cases in the theory of fuzzy sets).

*Interpretation:* the values of  $\mu_A(x)$  are interpreted as degrees of membership of  $x$  to  $A$  (0 denotes non-membership, 1 denotes membership, and  $1/2$  could represent semi-membership).

EXAMPLE 10.5.– Consider the property “to be much greater than 1”, defined on  $\mathbb{R}^+$ . It can be assigned to a fuzzy set  $A$  with  $\mu_A : \mathbb{R}^+ \rightarrow [0, 1]$ , where  $\mu_A$  is an arbitrary function that is continuous and non-decreasing such that, for example:

$$\mu_A(5) = 0.01$$

$$\mu_A(10) = 0.1$$

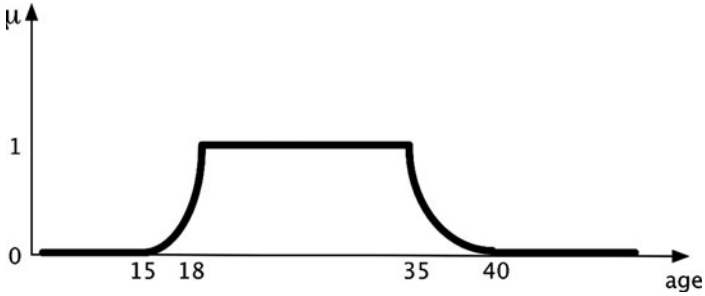
$$\mu_A(100) = 0.95$$

$$\mu_A(1000) = 1$$

⋮

□

EXAMPLE 10.6.– The characteristic function of the set of people who can be professional football players could be:



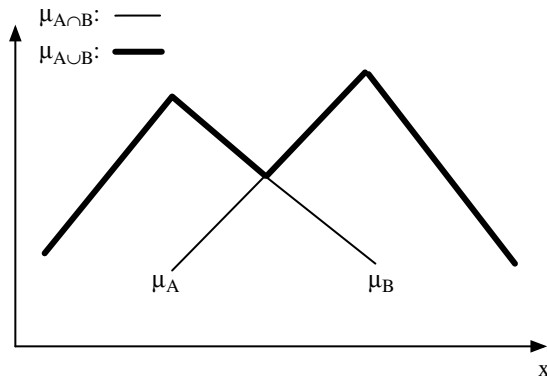
□

The usual operations are (in general) defined as follows.

DEFINITION 10.5.– (set operations on fuzzy sets).

- A fuzzy set  $A$  is empty iff  $\forall x \in U \mu_A(x) = 0$ .
- Two fuzzy sets  $A$  and  $B$  are equal (denoted by  $A = B$ ) iff  $\forall x \in U \mu_A(x) = \mu_B(x)$ .
- $A \subseteq B$  iff  $\forall x \in U \mu_A(x) \leq \mu_B(x)$   
(if an element is somehow an element of  $A$ , then it must be an element of  $B$  in at least the same degree).
- $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$  (for all  $x$ , we shall thus write  $\mu_{\bar{A}} = 1 - \mu_A$ ).
- $\mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$  (for all  $x$ , we shall thus write  $\mu_{A \cup B} = \max\{\mu_A, \mu_B\}$ ).
- $\mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$  (for all  $x$ , we shall thus write  $\mu_{A \cap B} = \min\{\mu_A, \mu_B\}$ ).

EXAMPLE 10.7.–



□

EXERCISE 10.1.– (properties). Prove that if  $A$ ,  $B$ , and  $C$  are fuzzy sets:

- a)  $A \cup B$  is the smallest set containing  $A$  and  $B$ .
- b)  $A \cap B$  is the biggest set contained in  $A$  and  $B$ .
- c)  $A \cup (B \cap C) = (A \cup B) \cap C$
- d)  $A \cap (B \cup C) = (A \cap B) \cup C$
- e)  $\overline{(A \cup B)} = \overline{A} \cap \overline{B}$
- f)  $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$
- g)  $C \cup (A \cap B) = (C \cup A) \cap (C \cup B)$
- h)  $C \cap (A \cup B) = (C \cap A) \cup (C \cap B)$

□

DEFINITION 10.6.– A fuzzy  $n$ -ary relation in  $E$  is a fuzzy set in  $E^n$ .

EXERCISE 10.2.– Give the generalized characteristic function for the fuzzy relation in  $\mathbb{N}$  (i.e.  $(x, y) \in \mathbb{N}^2$ ):  $x$  is much smaller than  $y$ . □

EXERCISE 10.3.– (relations). The composition of two binary relations  $R$  and  $S$ , denoted by  $R \circ S$ , is defined as follows.

$$x(R \circ S)y \text{ iff } \exists z \text{ such that } xRz \text{ and } zSy.$$

a) How would you define the composition of two *fuzzy binary relations*, i.e. what would the generalized characteristic function  $\mu_{R \circ S}$  be (as a function of  $\mu_R$  and  $\mu_S$ )?

How would you define the generalised characteristic function of a *fuzzy relation*  $R$  that is:

- b) *reflexive*?
- c) *symmetric*?

d) *antisymmetric*?

e) *transitive*? □

The most commonly used basic logic for FL is Łukasiewicz's  $L_{\aleph_1}$  (i.e. the set of truth values is the real interval  $[0, 1]$ ). To the qualifiers *true*, *very true*, *more or less true*, *neither true nor false*, etc. fuzzy sets of the interval  $[0, 1]$  will be assigned. This choice is necessary to be able to manipulate the logic practically. In general, we are interested in a finite (small) number of truth values.

These ideas were formalized by L. Zadeh, with the concept of *linguistic variables*.

More precisely, a *linguistic variable* is a 5-tuple:

$$\langle \mathcal{X}, \tau(X), U, G, M \rangle$$

where:

- $\mathcal{X}$  is the name of the variable (i.e. age, size, etc. from a *qualitative* point of view, by opposition to a *quantitative* point of view, with values 32 years, 1.75 m, etc.).
- $\tau(X)$  is the set of linguistic variables (young, not young, very young, not very young, etc.).
- $U$  is the universe of discourse (i.e. the set of values that the variable may take).
- $G$  is the set of production rules that generate  $\tau(X)$ .
- $M$  is the semantic rule.

$$M : \tau(X) \longrightarrow \mathcal{P}_{fuzzy}(U)$$

where  $\mathcal{P}_{fuzzy}(U)$  : set of parts of  $U$ , i.e. the set of all fuzzy subsets of  $U$ .

The idea behind the concept of linguistic variables is to obtain the fuzzy value of different terms as a function of those terms that were chosen as basic terms.

EXAMPLE 10.8.– (L. Zadeh). Linguistic variable: size

Basic term: small

$\tau(X) = \{\text{small, not small, very small, very (not small), not very small, very very small, etc.}\}$

is generated by the grammar (with axiom  $S$ ):

$$S \rightarrow A$$

$$S \rightarrow \text{not } A$$

$$A \rightarrow B$$

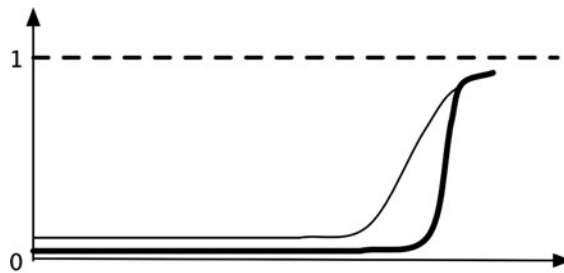
Number	Grammar rules	Semantics
1.	$S \rightarrow A$	$S_g = A_d$
2.	$S \rightarrow S \text{ or } A$	$S_g = S_d \cup A_d$
3.	$A \rightarrow B$	$A_g = B_d$
4.	$A \rightarrow A \text{ and } B$	$A_g = A_d \cap B_d$
5.	$B \rightarrow C$	$B_g = C_d$
6.	$B \rightarrow \text{not } C$	$B_g = \overline{C_d}$
7.	$C \rightarrow (S)$	$C_g = S_d$
8.	$C \rightarrow D$	$C_g = D_d$
9.	$C \rightarrow E$	$C_g = E_d$
10.	$D \rightarrow \text{very } D$	$D_g = (D_d)^2$
11.	$E \rightarrow \text{very } E$	$E_g = (E_d)^2$
12.	$D \rightarrow \text{true}$	$D_l = \text{true}$
13.	$E \rightarrow \text{false}$	$E_l = \text{false}$

$B \rightarrow \text{very } B$   
 $B \rightarrow (S)$   
 $B \rightarrow \text{small}$

If we assume that the generalized characteristic function (see definition 10.4) of the extension A of a concept is  $\mu_A$ , then in general, we take (other similar choices are of course possible):

$$\mu_{\text{very}-A}(x) = (\mu_A(x))^2$$

graphical meaning:



tall : —————  
 very tall : —————

$$\mu_{\text{not}-A}(x) = 1 - \mu_A(x)$$

If the grammar had authorized it, we could have let, for example:

$$\mu_{\text{more-or-less}-A}(x) = \sqrt{\mu_A(x)}$$

□



EXAMPLE 10.9.– (syntax and semantics). The symbols occurring in the rules on the right-hand side correspond to the fuzzy subsets of  $[0, 1]$ . Of course, indices  $l$  and  $r$  correspond to *left* and *right* (of a rule).  $\square$

EXERCISE 10.4.– Does the grammar of example 10.9 permit us to generate the linguistic variable *not very true and not very false*? If so, compute its truth value.  $\square$

Fuzzy sets permit us to obtain an elegant solution to the paradox<sup>4</sup>.

All men are small.

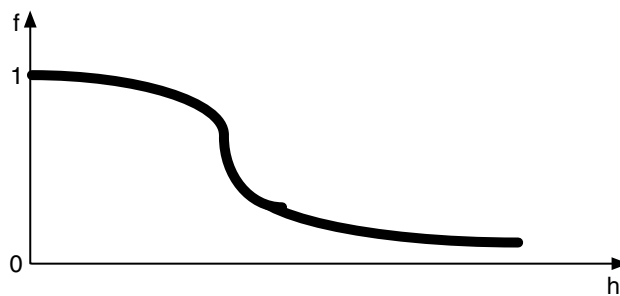
The only remark to be made is that not all generalized characteristic functions correspond to the common notion of “small”. We define:

$$\mu_X(x) = f(h(x))$$

for some function  $f : X \rightarrow P$ , which, to represent the extension of the concept “small man” must be<sup>5</sup>:

- 1) continuous (our appreciation changes progressively);
- 2) monotonically decreasing;
- 3)  $f(h(x)) = 1$  (or  $f(h(x)) \approx 1$ ), for those men that are certainly small;
- 4)  $f(h(x)) = 0$  (or  $f(h(x)) \approx 0$ ) for those men that are certainly not small.

For example:



<sup>4</sup> The solution is given by J. Goguen.

<sup>5</sup> The required *properties* on generalized characteristic functions are a key point in the formalization of FLs.

We define the fuzzy relation “smallness of  $y$  with respect to  $x$ ”

$$H_X(x, y) = \frac{\mu_X(y)}{\mu_X(x)} = \frac{f(h(y))}{f(h(x))}$$

$$\mu_X(y) = H_X(x, y) \times \mu_X(x), \text{ for } h(y) \geq h(x)$$

we assume there exists an  $x \in X$  such that  $\mu_X(x) = 1$  (there is a man who is certainly small), for example,  $h(x) \approx 0$  (i.e. his height is almost zero).

Consider  $y \in X$  such that  $h(y) \geq h(x)$ , we may construct a finite sequence  $x = x_0, x_1, x_2, \dots, x_{N-1}, x_N = y$  and:

$$\mu_X(y) = \prod_{x=1}^N H_X(x_{i-1}, x_i)$$

This formula translates the intuition that, as the process is iterated, the conviction that  $y$  is small disappears. Indeed, the product of the values in  $]0, 1[$  gets closer and closer to 0. The mathematical details are simple and are left to the reader.

REMARK 10.5.– We may see the origin of the paradox of small men (of the heap of grains, etc.) in the fact that each time a new man is considered, the memory that in general there were many men taller than someone small was lost. The treatment proposed above *restores* this memory.  $\square$

### 10.2.1. Inference in FL

L. Zadeh proposed to treat the gradual membership of element  $x$  to a fuzzy set  $A$  by fixing values  $\alpha, \beta$ ;  $0 < \alpha < \beta < 1$  and to use as a convention:

if  $\mu_A(x) \leq \alpha$ , then  $x$  does not belong to  $A$ ;

if  $\beta \leq \mu_A(x)$ , then  $x$  belongs to  $A$ ;

if  $\alpha < \mu_A(x) < \beta$ , then the membership of  $x$  to  $A$  is undetermined.

From the point of view of the corresponding logic, this would lead to a logic with three truth values: **T**, **F**, **i** (see section 10.1).

The method proposed above can be viewed as a generalization of this simple idea.

It is natural to try to apply inference rules from classical logic to non-classical logics. For example, the resolution method was extended (by C. Chang) more than 30 years ago from FOL to first-order FL. But first we must formally define the semantics of FL.

10.2.1.1. *Syntax*

The same as for classical logic.

10.2.1.2. *Semantics*

The only difference with FOL (that corresponds to intuition) is that every predicate  $P^{(n)}$  is not interpreted as an  $n$ -ary relation (i.e. the *set* of  $n$ -tuples of the relation) in the universe of discourse, but as a *fuzzy set*, or more precisely the corresponding generalized characteristic set (in other words, to every  $n$ -tuple  $(x_1, \dots, x_n)$ , we do not assign **true** or **false** to  $P^{(n)}(x_1, \dots, x_n)$ , but a value in  $[0, 1]$ , which is given by the characteristic function).

It is thus possible to view classical logic as a particular case of FL.

The semantics (i.e. the value given to) composed wffs is given by the function  $v$  defined below.

DEFINITION 10.7.–

- $v(\neg \mathcal{F}) = 1 - v(\mathcal{F})$
- $v(\mathcal{F}_1 \wedge \mathcal{F}_2) = \min[v(\mathcal{F}_1), v(\mathcal{F}_2)]$
- $v(\mathcal{F}_1 \vee \mathcal{F}_2) = \max[v(\mathcal{F}_1), v(\mathcal{F}_2)]$
- $v(\mathcal{F}_1 \Rightarrow \mathcal{F}_2) = \min[1, 1 - v(\mathcal{F}_1) + v(\mathcal{F}_2)]$

or:

- $v(\mathcal{F}_1 \Rightarrow \mathcal{F}_2) =$   
 $1$  if  $v(\mathcal{F}_1) \leq v(\mathcal{F}_2)$   
 $v(\mathcal{F}_2)$  if  $v(\mathcal{F}_1) > v(\mathcal{F}_2)$
- $v(\forall x \mathcal{F}(x)) = \inf[v(\mathcal{F}(x)) \mid x \in D]$
- $v(\exists x \mathcal{F}(x)) = \sup[v(\mathcal{F}(x)) \mid x \in D]$

An interpretation  $\mathcal{I}$  satisfies (respectively, falsifies) a formula  $\mathcal{F}$  iff  $v(\mathcal{F}) \geq 0.5$  (respectively  $< 0.5$ ).

REMARK 10.6.– In general,

$$v(A \wedge \neg A) \neq 0$$

$$v(A \vee \neg A) \neq 1.$$

□

EXAMPLE 10.10.– Consider the wff  $\forall x\exists yP(x, y)$

on domain  $D = \{a_1, a_2\}$

$$v(\exists yP(x, y)) = \max([v(P(x, a_1)), v(P(x, a_2))])$$

$$v(\forall x\exists yP(x, y)) = \min[\max[v(P(a_1, a_1)), v(P(a_1, a_2))],$$

$$\max[v(P(a_2, a_1)), v(P(a_2, a_2))]] \quad \square$$

The next step is to find a universe on which it is sufficient to focus to test whether a wff is contradictory.

### 10.2.2. Herbrand's method in FL

The key property (see theorem 5.7) is that given a set of clauses  $S$ , we can associate to any interpretation  $\mathcal{I}$  of  $S$  a Herbrand interpretation  $\mathcal{I}_H$  such that if  $\mathcal{I} \models S$  then  $\mathcal{I}_H \models S$ .

We show this on an example.

EXAMPLE 10.11.– Consider the clause:

$$\forall xP(a, f(x))$$

and the interpretation  $\mathcal{I}$ :

$$D = \{1, 2\}$$

$$a \mapsto 1$$

$$f(1) = 2$$

$$f(2) = 1$$

$$v(P(1, 1)) = 0.6$$

$$v(P(1, 2)) = 0.7$$

$$v(P(2, 1)) = 0.2$$

$$v(P(2, 2)) = 0.3$$

This interpretation is a model of  $\forall xP(a, f(x))$ . Indeed,  $v(\forall xP(a, f(x))) = \min[P(1, 2), P(1, 1)] = \min[0.7, 0.6] = 0.6$

The corresponding interpretation  $\mathcal{I}_H$  is:

$$v(P(a, f(a))) = v(P(1, f(1))) = v(P(1, 2)) = 0.7$$

$$v(P(a, f^2(a))) = v(P(1, f^2(1))) = v(P(1, 1)) = 0.6$$

$$v(P(a, f^3(a))) = v(P(1, f^3(1))) = v(P(1, 2)) = 0.7$$

⋮

□

Semantic trees are defined in the same way as for FOL, by replacing  $L$  with  $v(L) \geq 0.5$  and  $\neg L$  by  $v(L) < 0.5$ .

#### 10.2.2.1. Resolution and FL

In PL, the resolvent is a logical consequence of the parent clauses (i.e. every model of the parent clauses is a model of the resolvent).

We first analyze an example the way the resolution rule for PL transmits truth values when it is applied to FL.

EXAMPLE 10.12.– Consider the two following clauses:

$$C_1: \neg P \vee Q$$

$$C_2: P$$

(whose resolvent is  $Q$ )

with:

$$v(P) = 0.3$$

$$v(Q) = 0.2$$

$$v(C_1) \wedge v(C_2) = v((\neg P \vee Q) \wedge P) = \min[\max[v(\neg P), v(Q)], v(P)] = \min[0.7, 0.3] = 0.3$$

hence, there is a difference with classical PL:

$$v(Q) = 0.2 < v(C_1 \wedge C_2)$$

□

This example is a particular case of the following theorem.

**THEOREM 10.1.**– *Let  $C_1$  and  $C_2$  denote two clauses and  $\mathcal{R}(C_1, C_2)$  denote a resolvent.*

*If  $\max[v(C_1), v(C_2)] = b$  and  $\min[v(C_1), v(C_2)] = a > 0.5$*

*then:*

$$a \leq v(\mathcal{R}(C_1, C_2)) \leq b$$

**PROOF.**–  $C_1: P \vee \alpha, C_2: \neg P \vee \beta$  ( $\alpha$  and  $\beta$ : disjunctions of literals)

$$\mathcal{R}(C_1, C_2) : \alpha \vee \beta$$

Without loss of generality, we assume:

\*  $a \leq b$  and:

$$v(C_1) = \max[v(P), v(\alpha)] = a \quad (1)$$

$$v(C_2) = \max[v(\neg P), v(\beta)] = b \quad (2)$$

from (1) and (2) and by definition of max:

$$v(\alpha) \leq a$$

$$v(\beta) \leq b$$

there are two cases to consider:

i)  $v(\alpha) = a$

$$v(\mathcal{R}(C_1, C_2)) = v(\alpha \vee \beta) = \max[v(\alpha), v(\beta)] = \max[a, v(\beta)]$$

$$a \leq \mathcal{R}(C_1, C_2) \leq b \text{ (see *)}$$

ii)  $v(\alpha) < a$ , from (1)  $v(P) = a$

By hypothesis  $a > 0.5$ , hence  $v(\neg P) = 1 - v(P) < 0.5 < a$

from (2) and \*  $v(\beta) = b \geq a$

$$v(\mathcal{R}(C_1, C_2)) = v(\alpha \vee \beta) = \max[v(\alpha), v(\beta)] = b$$

$$a \leq \mathcal{R}(C_1, C_2) \leq b$$

This theorem can easily be generalized. □

**THEOREM 10.2.**– *Consider the set of clauses  $S = \{C_1, \dots, C_k\}$  ( $k \geq 2$ )*

*If  $\max[v(C_1), \dots, v(C_k)] = b$ ,  $\min[v(C_1), \dots, v(C_k)] = a$  and  $C \in \mathcal{R}_S^n$  ( $n \geq 0$ )*

*then*

$$a \leq v(C) \leq b.$$

### 10.3. Modal logics

In a dictionary on language sciences (O. Ducrot and T. Todorov), it is written:

Logicians and linguists have often deemed it necessary to distinguish, in an enunciation act, a representative content, sometimes called *dictum* (relating a predicate to a subject), and an attitude expressed by the subject speaking of the content, that is the *modus* or *modality*.

The four following sentences have the same *dictum*, but different modalities:

- Peter will come.
- Let Peter come!
- It is possible that Peter will come.
- Peter must come.

Modal logic studies *modalities*, i.e. the logical operations that *qualify* the assertions (or more precisely that qualify their *truth modes*). We will say, for example, that a proposition is *necessarily* true, or *possibly* true, or that it *has always been* true, or that it *will be true one day*, or that *we know* that it is true, or that it *should* be true, etc.

Originally, modal logic was designed to characterize the *possible* and *necessary* truths. It was progressively extended to the study of statements on knowledge, beliefs, time, ethics, properties of programs, etc.

They are particularly relevant in AI, for example, in the so-called description logics for ontologies and in multi-agent systems.

The concepts of *necessity* and of *possibility* have been studied by philosophers, ever since Aristotle, at least.

It is generally acknowledged that C.I. Lewis (in the 1930s) first studied these *modalities*. His analysis was mainly concerned with *strict implication*, thus called to be distinguished from *material implication* and its paradoxes (see exercise 3.7).

*A strictly implies B* (written  $A \multimap B$ ) iff  $A \wedge \neg B$  is *impossible*, or, with modern notations,

$$\neg \diamond(A \wedge \neg B) \text{ or, equivalently, } \Box(A \Rightarrow B)$$

where  $\diamond$  is interpreted as *possible* and  $\Box$  is interpreted as *necessary*.

Little by little, the study of the connectives  $\diamond$  and  $\Box$  themselves became a field by itself.

For example, in computer science, when referring to a non-deterministic program,  $\Box A$  means “in every execution that halts,  $A$  is **T**”, whereas  $\Diamond A$  means “there exists an execution that halts with  $A$  **T**”.

Gödel was interested in modalities from the point of view of *provability*.

For historical purposes, i.e. Aristotle’s philosophy and its huge influence on Western culture, those that were the most studied are:

- ontic modalities: it is *necessary* (*possible*, *impossible*, ...) that...
- epistemic modalities: Peter *knows* that...
- temporal modalities: since, starting from, until, he *always* thinks that...
- deontic modalities: it is forbidden, it is permitted, it is illegal...

Other modalities:

- doxastic modalities: we believe that,...
- metalogic modalities: it is provable that, this is satisfiable,...

There exist propositional, first-order and higher-order modal logics. We restrict ourselves to propositional modal logics (we will see they have a great expressive power).

One essential characteristic of these logics is the fact that their connectives are not truth functional.

EXAMPLE 10.13.– (See example 2.6.)

Consider the conjecture (we could have chosen any other one)

$P = NP$

It could be **T** or (exclusive or) **F**, but the propositions:

$\Diamond (P = NP)$ : It is possible that  $P = NP$

$\Diamond (\neg (P = NP))$ : It is possible that  $P \neq NP$

could both be considered as true propositions. We thus have (no matter the value given to  $P = NP$ ):

$\Diamond \mathbf{T} = \mathbf{T}$

(\*)  $\Diamond \mathbf{F} = \mathbf{T}$



On the other hand:

$1 + 1 = 3$  is **F** and it is normal to consider  $\diamond(1 + 1 = 3)$  as **F**. In other words, we do not imagine that it is possible for  $1 + 1 = 3$  to be **T**.

By replacing **F** in (\*) by  $1 + 1 = 3$  (i.e. by **F**), we obtain:

$$(*) (*) \diamond \mathbf{F} = \mathbf{F}$$

By considering propositions (\*) and (\*) (\*), we verify that  $\diamond$  (connective meaning *possibly*) is not truth functional.  $\square$

### 10.3.1. *Toward a semantics*

There exist relational (*possible worlds*), algebraic, and topological semantics.

We will study the semantics that is best known: the semantics of *possible worlds*.

The notion of possible worlds seems to have been around at least since Leibniz. Wittgenstein used to talk of the *possible state of things*.

In a book that is considered as a reference in logic (and that ignores modal logic)<sup>6</sup>, it is written:

“... Therefore, a ‘possible world’, or model of [a language]  $\mathcal{L}$  ...”

This seems to show that the transition from one to the other (i.e. from classical logic to non-classical ones) can be made naturally via the notion of possible worlds.

To better see the relationship between the notion of a model in classical logic and that of a possible world, we consider the formula:  $[(A \vee B) \Rightarrow C] \Leftrightarrow [(A \Rightarrow C) \vee (B \Rightarrow C)]$

and the interpretations:

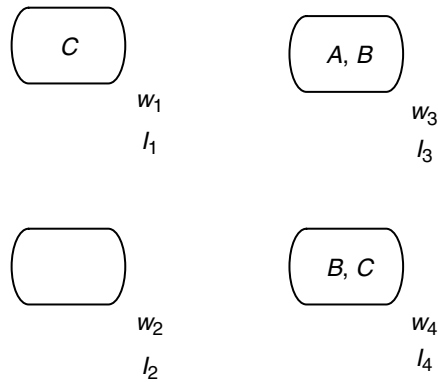
$$I_1 = \{C\}, I_2 = \emptyset, I_3 = \{A, B\} \text{ and } I_4 = \{B, C\}$$

We can imagine that each  $I_i$  ( $1 \leq i \leq 4$ ) corresponds to a situation, a moment, a state, a state of knowledge, a world, etc.

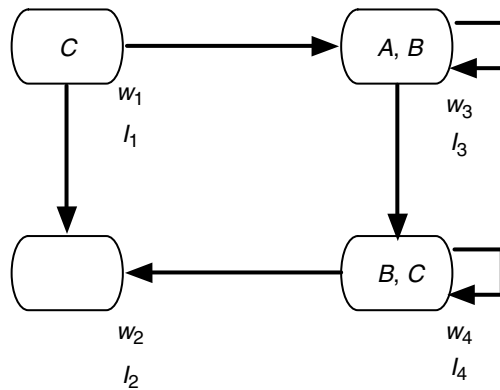
---

<sup>6</sup> Chang and Keisler: *Model Theory*.

These interpretations can be represented as worlds (we only provide the elementary propositions that are evaluated to true).



We can also add a relation between these worlds (which leads to the so-called Kripke semantics or the possible worlds semantics).



And instead of saying:

(CL) Formula  $\varphi$  is satisfied (or not satisfied) by interpretation  $I$

we will say:

(NCL) Formula  $\varphi$  is satisfied (or not satisfied) in the world  $w_i$  (which implicitly contains an interpretation  $I$ )

It will also be possible to construct formulas mentioning different worlds.

REMARK 10.7.– The notion of possible worlds is familiar (not necessarily with the same name) in the study of probability theory. For example, when analyzing the logical possibilities of the results of an arbitrary experiment (for example, throwing a die).  $\square$

More formally.

10.3.1.1. *Syntax (language of modal logic)*

$$\langle \text{formula} \rangle ::= \langle \text{atomic formula} \rangle \mid \neg \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \wedge \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \vee \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \Rightarrow \langle \text{formula} \rangle \mid \langle \text{formula} \rangle \Leftrightarrow \langle \text{formula} \rangle \mid \square \langle \text{formula} \rangle \mid \diamond \langle \text{formula} \rangle$$

$$\langle \text{atomic formula} \rangle ::= P_i (i \geq 0)^7$$

or more concisely:

$$\varphi ::= P_i \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \Rightarrow \varphi_2 \mid \varphi_1 \Leftrightarrow \varphi_2 \mid \square\varphi \mid \diamond\varphi$$

$$\diamond\varphi \stackrel{\text{def}}{=} \neg\square\neg\varphi$$

One of the characteristics of modal logics is their “parametrized flexibility”. This sentence, whose meaning will be made accurate in what follows, is metaphorical and expresses the fact that the new connectives that are introduced by these logics can be used to express very different characteristics in very different domains. For example, consider the different meanings of formula  $\square\varphi$ :

- $\square\varphi$ : necessarily  $\varphi$
- $\square\varphi$ : in the future always  $\varphi$
- $\square\varphi$ : it must be the case that  $\varphi$
- $\square\varphi$ : we know that  $\varphi$
- $\square\varphi$ : we believe that  $\varphi$
- $\square\varphi$ : after any execution of the program has halted, we have  $\varphi$
- $\square\varphi$ : in Peano’s arithmetic we can prove  $\varphi$

---

<sup>7</sup> It is worth mentioning that Aristotle was the first to use propositional variables, i.e. letters that can be replaced by *propositional* symbols and not by terms (of a syllogism, see definition 2.8) in modal logic.

10.3.1.2. *Semantics*

DEFINITION 10.8.– (Kripke semantics, Kripke models).

- A modal frame or *K-modal frame* is a pair  $\mathcal{F} = (W, R)$  where  $W$  is a non-empty set and  $R$  is a binary relation on  $W$ .
- A Kripke model<sup>8</sup> is a triple:

$$\mathcal{M} = (W, R, v)$$

with:

- $W$ : non-empty set (of worlds);
- $R$ : binary relation on  $W$  (accessibility relation);
- $v : W \rightarrow 2^{\mathcal{P}}$  or  $v : \mathcal{P} \rightarrow 2^W$  ( $\mathcal{P}$ : set of all atomic propositions (formulas).  $2^{\mathcal{P}}$  and  $2^W$ , respectively, denote the sets of parts of  $\mathcal{P}$  and  $W$ );
- $v$  is called the interpretation function or valuation ( $v$  indistinctly provides the set of worlds in which a proposition is evaluated to  $\mathbf{T}$  or the set of propositions that are evaluated to  $\mathbf{T}$  in each world)
- we shall indistinctly say  $w \in W$  or  $w \in \mathcal{M}$ ;
- the frame  $\mathcal{F}$  is called the base of the model.
- The domain of  $v$  is extended to the set of formulas ( $\varphi$  is an arbitrary formula)  $v(\varphi) = \{w \mid \mathcal{M}, w \models \varphi\}$ .

DIGRESSION 10.1.– Kripke defined propositions as functions from the set of worlds to the truth values and an  $n$ -ary predicate as a function from the set of worlds to the set of  $n$ -ary relations. □

DEFINITION 10.9.– (satisfiability, validity). *In what follows,  $\varphi$  is a wff and  $Mod$  is a set of Kripke models  $\mathcal{M} = (W, R, v)$ .*

$\mathcal{M}, w \models \varphi$  means: the formula  $\varphi$  is satisfied in the world  $w$  of the model  $\mathcal{M}$ . The relation  $\models$  is defined as follows ( $Mod$  denotes a set of models):

- $\mathcal{M}, w \models P$  (sometimes we write  $\mathcal{M} \models P(w)$ ) iff  $P \in v(w)$  for  $P \in \mathcal{P}$ ;
- $\mathcal{M}, w \models \neg\varphi$  iff not( $\mathcal{M}, w \models \varphi$ );
- $\mathcal{M}, w \models \varphi_1 \wedge \varphi_2$  iff  $\mathcal{M}, w \models \varphi_1$  and  $\mathcal{M}, w \models \varphi_2$ ;
- $\mathcal{M}, w \models \varphi_1 \vee \varphi_2$  iff  $\mathcal{M}, w \models \varphi_1$  or  $\mathcal{M}, w \models \varphi_2$ ;
- $\mathcal{M}, w \models \varphi_1 \Rightarrow \varphi_2$  iff not( $\mathcal{M}, w \models \varphi_1$ ) or  $\mathcal{M}, w \models \varphi_2$ ;
- $\mathcal{M}, w \models \varphi_1 \Leftrightarrow \varphi_2$  iff  $\mathcal{M}, w \models \varphi_1 \Rightarrow \varphi_2$  and  $\mathcal{M}, w \models \varphi_2 \Rightarrow \varphi_1$ ;

---

<sup>8</sup> Note that, in accordance with usage, we talk about *Kripke models* and not Kripke interpretations. Thus, a formula may be evaluated to  $\mathbf{F}$  in a Kripke model. We may also talk of the Kanger–Kripke semantics.

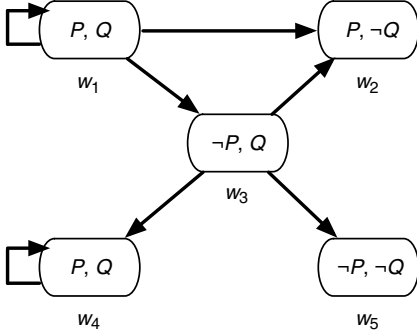
- $\mathcal{M}, w \models \Box\varphi$  iff for all  $w'$  such that  $R(w, w')$ :  $\mathcal{M}, w' \models \varphi$ ;  
 $\Box\varphi$ : necessarily  $\varphi$ , or  $\varphi$  is necessarily **T**<sup>9</sup>;
- $\mathcal{M}, w \models \Diamond\varphi$  iff there exists  $w'$  such that  $R(w, w')$ :  $\mathcal{M}, w' \models \varphi$ ;  
 $\Diamond\varphi$ :  $\varphi$  is possible or  $\varphi$  is **T** in a possible (accessible) world;
- $\varphi$  is Mod – satisfiable iff there exists  $\mathcal{M} \in \text{Mod}$  and  $w \in \mathcal{M}$  such that  $\mathcal{M}, w \models \varphi$ ;
- $\varphi$  is valid in model  $\mathcal{M}$  ( $\mathcal{M} \in \text{Mod}$ ), written  $\mathcal{M} \models \varphi$  iff  $v(\varphi) = W$  (i.e. iff for all  $w \in W$   $\mathcal{M}, w \models \varphi$ );
- $\varphi$  Mod – valid iff  $\varphi$  is valid in all models in Mod;
- $\varphi$  is valid in modal frame  $(W, R)$  iff  $\varphi$  is valid in all models admitting  $(W, R)$  as a base;
- A model is finite iff the set of worlds of the modal frame is finite; it is infinite otherwise<sup>10</sup>.

REMARK 10.8.– The underlying notion of necessity in the definition is not always related to a necessity in the *real world*, as we see it. There could be, for example, several parallel worlds (or flows of time) that coexist (that flow at different speeds) and we only have access to one of them. □

We obtain different logics by imposing conditions (in particular none) on the accessibility relation. The best-known logics are K, S5, S4 etc.

These logics can also be characterized by formal systems.

EXAMPLE 10.14.– Consider the Kripke model represented by the graph below. We indicate the corresponding interpretation in each world. The edges of the graph define the accessibility relation.



<sup>9</sup> A necessary truth is a truth that is verified in all possible (and accessible) worlds.  
<sup>10</sup> As in the case of FOL, the property of admitting finite models (see definition 5.15) is important for the design of decision procedures for modal logics.

In this model  $\mathcal{M}$ , we have:

- $\mathcal{M}, w_1 \models \Box(P \vee Q)$
- *not*  $(\mathcal{M}, w_3 \models \Box(P \vee Q))$
- *not*  $(\mathcal{M}, w_3 \models \Diamond(\neg P \wedge Q))$
- $\mathcal{M}, w_3 \models \Diamond(\neg P \wedge \neg Q)$  □

### 10.3.2. How to reason with modal logics?

Similar to classical logic, there are different approaches. We shall study the following ones:

- syntactic approach (formal systems);
- direct approach;
- translation approach.

In the first approach, we provide a formal system (see section 3.3) for the logic under consideration. In the direct approach, we give inference rules for the modal connectives, and in the translation approach, we translate formulas into FOL and reason in FOL<sup>11</sup>.

We present here the syntactic approach and the translation approach for some propositional modal logics. The direct approach is treated for temporal logics, which can be considered as particular cases of modal logics.

#### 10.3.2.1. Formal systems approach

We specify those that are best known. *The language is still the one of section 10.3.1.1.*

DEFINITION 10.10.–

– *The minimal logic K (K as Kripke)*

$\mathcal{A}$  (axioms):

All tautologies of PL<sup>12</sup>  $\cup \Box(P \Rightarrow Q) \Rightarrow (\Box P \Rightarrow \Box Q)$ .

$\mathcal{R}$  (inference rules):

MP:  $\frac{P \quad P \Rightarrow Q}{Q}$     necessitation :  $\frac{P}{\Box P}$     substitution    % see remark 3.20

<sup>11</sup> Previously, the founder of temporal logic, appears to be a pioneer of the translation method.

<sup>12</sup> Note that we could have chosen any complete formal system, for example,  $\mathcal{S}_1$  (see section 3.4) for PL. The choice that was made permits us not to have to (re)prove all theorems (tautologies) of PL.

## Other logics

- Inference rules: the same
- Axioms:
  - $T = K \cup \{\Box P \Rightarrow P\}$
  - $D = K \cup \{\Box P \Rightarrow \Diamond P\}$
  - $S4 = K \cup \{\Box P \Rightarrow P, \Box P \Rightarrow \Box \Box P\}$
  - $S5 = S4 \cup \{\Diamond \Box P \Rightarrow P\}$
  - $G = K \cup \{\Box(\Box P \Rightarrow P) \Rightarrow \Box P\}$

EXERCISE 10.5.– Give the proofs (deductions) that correspond to the following theorems:

- a)  $\vdash_T P \Rightarrow \Diamond P$
- b)  $A \Rightarrow B \vdash_K \Box A \Rightarrow \Box B$
- c)  $\vdash_{S5} \Diamond \Diamond P \Rightarrow \Diamond P$
- d)  $\vdash_{S5} \Box(P \vee Q) \Rightarrow (\Box P \vee \Box Q)$
- e)  $\vdash_K \Box(A \wedge B) \Rightarrow (\Box A \wedge \Box B)$
- f)  $\vdash_K \Box A \wedge \Box B \Rightarrow \Box(A \wedge B)$
- g)  $\vdash_G \Box P \Rightarrow \Box \Box P$
- h)  $\vdash_{S4} \Box A \vee \Box B \Rightarrow \Box(\Box A \vee \Box B)$  □

## 10.3.2.2. Translation approach

The connectives that are difficult to translate are  $\Box$  and  $\Diamond$ . The translation function (Andréka, Németi, and Van Benthem)  $\text{tr}$  (see definition 10.9) is defined as follows:

$\begin{aligned} \text{tr}(P) &: P(X) \\ \text{tr}(\neg\varphi) &: \neg\text{tr}(\varphi) \\ \text{tr}(\varphi \vee \psi) &: \text{tr}(\varphi) \vee \text{tr}(\psi) \\ \text{tr}(\varphi \wedge \psi) &: \text{tr}(\varphi) \wedge \text{tr}(\psi) \\ \text{tr}(\varphi \Rightarrow \psi) &: \text{tr}(\varphi) \Rightarrow \text{tr}(\psi) \\ \text{tr}(\Box\varphi) &: \forall y(R(X, y) \Rightarrow \text{tr}(\varphi(y))) \\ \text{tr}(\Diamond\varphi) &: \exists y(R(X, y) \wedge \text{tr}(\varphi(y))) \end{aligned}$
--

$P$ : propositional symbol.

$X$ : free variable, denoting the world in which we are.

$y$ : fresh variable not occurring in the formulas to be translated.

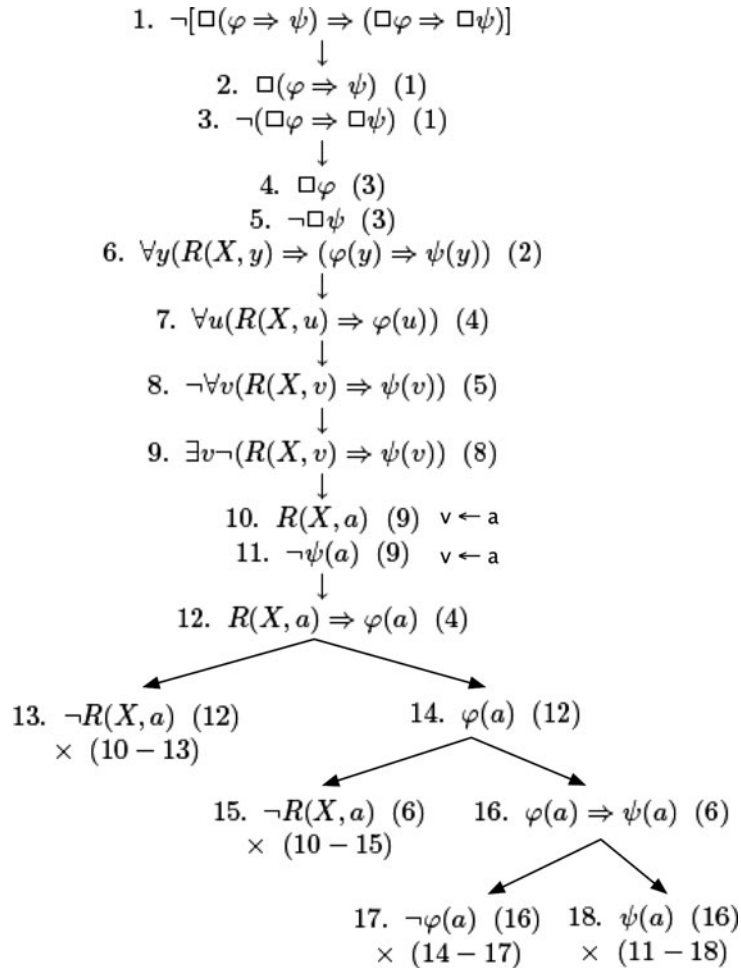
$R$ : accessibility relation.

The idea behind this translation is of course the capture of the semantics imposed by the definition:  $P(y)$  (respectively,  $\varphi(y)$ ) should be interpreted as  $P$  (respectively,  $\varphi$ ) is **T** in world  $y$ .

EXAMPLE 10.15.– The formula whose validity is to be proven is:

$$\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$$

This is the “specifically modal” axiom of logic K (see definition 10.10). Note that no particular property is imposed on the accessibility relation (this should be compared with exercise 10.7).





EXAMPLE 10.16.– Using the translation method and the method of semantic tableaux:

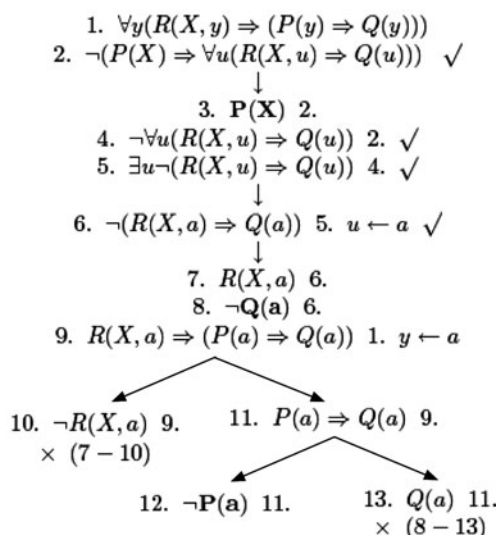
a) give a Kripke model  $\mathcal{M}$  such that in a world  $w_1$ :  $\mathcal{M}, w_1 \models \Box(P \Rightarrow Q)$  and  $\mathcal{M}, w_1 \not\models P \Rightarrow \Box Q$ ;

b) give a Kripke model  $\mathcal{M}$  such that in a world  $w_1$ :  $\mathcal{M}, w_1 \models P \Rightarrow \Box Q$  and  $\mathcal{M}, w_1 \not\models \Box(P \Rightarrow Q)$ .

a) We consider the set  $\{\Box(P \Rightarrow Q), \neg(P \Rightarrow \Box Q)\}$

Translation:

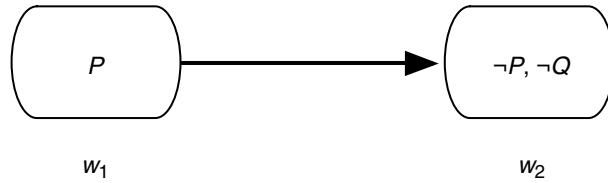
$$\begin{aligned} \Box(P \Rightarrow Q) &: \forall y(R(X, y) \Rightarrow \overbrace{(P(y) \Rightarrow Q(y))}^{P(y)}) \\ \neg(P \Rightarrow \Box Q) &: \neg(P(X) \Rightarrow \forall u(R(X, u) \Rightarrow Q(u))) \end{aligned}$$



REMARK 10.9.– The constants that are introduced by variable instantiations (here  $a$ ) correspond to accessible worlds.  $\square$

The branch going through leaf 12 cannot be closed by instantiating the only formulas that can still be instantiated.

We have constructed the desired model:

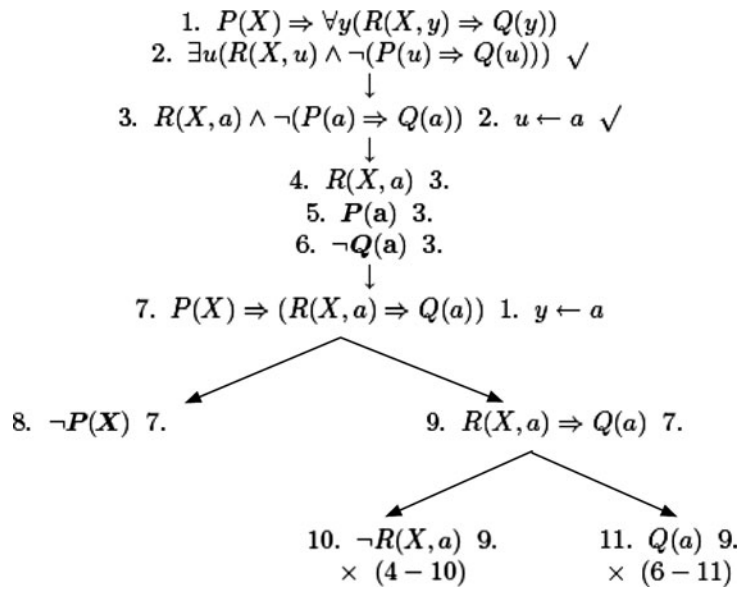


b) We consider the set  $\{P \Rightarrow \Box Q, \neg \Box(P \Rightarrow Q)\}$

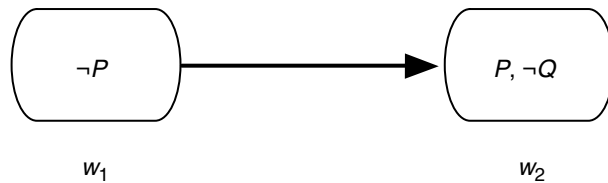
Translation:

$$P \Rightarrow \Box Q : P(X) \Rightarrow \forall y(R(X, y) \Rightarrow Q(y))$$

$$\neg \Box(P \Rightarrow Q) : \Diamond \neg(P \Rightarrow Q) : \exists u(R(X, u) \wedge \neg(P(u) \Rightarrow Q(u)))$$



We have constructed the desired model:



□

EXERCISE 10.6.– (sea battle argument). Express the following argument (given by Aristotle) in modal logic and verify its correctness using the translation approach.

(Similar to classical logic, given an argument expressed in a natural language, the classification of this argument as correct or incorrect will generally depend on the way it is translated.)

If I give the order to attack then necessarily there will be a sea battle tomorrow. Otherwise, necessarily, there will be none. I either give the order or I do not give it. Hence, necessarily, there will be a sea battle tomorrow, or necessarily there will be none.

Two classical translations of the sea battle argument are given below:

$P$ : “I give the order to attack”;

$Q$ : “There will be a sea battle tomorrow”.

$$\frac{\begin{array}{l} P \Rightarrow \Box Q \\ \neg P \Rightarrow \Box \neg Q \\ P \vee \neg P \end{array}}{\Box Q \vee \Box \neg Q}$$
  

$$\frac{\begin{array}{l} \Box(P \Rightarrow Q) \\ \Box(\neg P \Rightarrow \neg Q) \\ P \vee \neg P \end{array}}{\Box Q \vee \Box \neg Q}$$

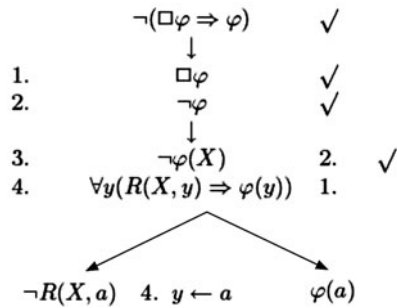
□

EXAMPLE 10.17.– (accessibility relation for  $T, S4, S5$ ). We can use the translation method and the method of semantic tableaux to try to discover the *sufficient* properties that are required of the accessibility relation so that the following axioms are valid.

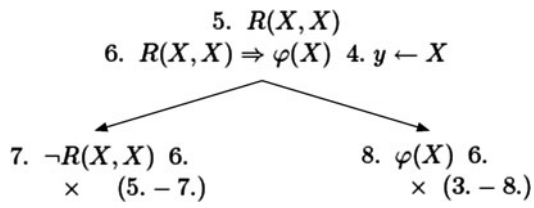
$$\Box \varphi \Rightarrow \varphi$$

$$\Diamond \Box \varphi \Rightarrow \varphi$$

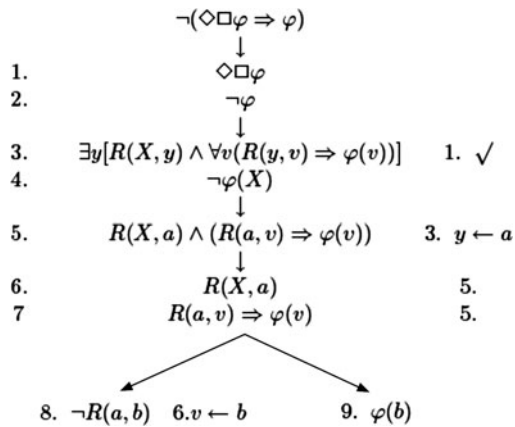
$$\Box \varphi \Rightarrow \Box \Box \varphi$$



If  $R$  is reflexive, i.e.  $\forall xR(x, x)$ , then we can graft to each branch of the tree:



and the two branches can be closed.



If  $R$  is symmetric, i.e.  $\forall x\forall yR(x, y) \Rightarrow R(y, x)$ , then we can graft to each branch of the tree:



As an example, we give here a proof of this property that is representative of the properties that appear in textbooks on this topic.

$\Box\varphi \Rightarrow \Box\Box\varphi$  is valid in a modal frame iff the accessibility relation  $R$  is transitive.

*if*

We show that if  $R$  is transitive and we have

$$\mathcal{M}, w_1 \models \Box\varphi$$

then we also have:

$$\mathcal{M}, w_1 \models \Box\Box\varphi$$

$R$  is transitive, i.e.:

$$R(w_1, w_2) \wedge R(w_2, w_3) \Rightarrow R(w_1, w_3)$$

hence, if formula  $\Box\varphi$  is satisfied in world  $w_1$ , then it is satisfied (see definitions 10.8 and 10.9) in all worlds that are *accessible* from (related to)  $w_1$ ; hence, it is satisfied in  $w_3$ , so that  $\Box(\Box\varphi)$  is also satisfied in  $w_1$ .

*only if*

We prove the contrapositive, we assume that  $R$  is not transitive and we need to prove that:

$$\mathcal{M}, w_1 \models \Box\varphi$$

and that:

$$\mathcal{M}, w_1 \not\models \Box\Box\varphi$$

to do so, we *propose*, for example,  $v(w_2) = \varphi$ ;  $v(w_3) = \neg\varphi$ .

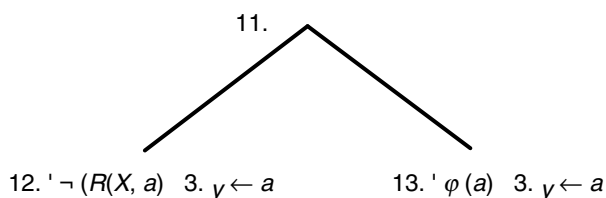
This valuation is possible as  $R$  is not transitive; hence, we may assume that we have  $\neg\varphi$  in  $w_3$ . This would not be possible if  $R$  had been transitive, as we assumed:

$$\mathcal{M}, w_1 \models \Box\varphi$$

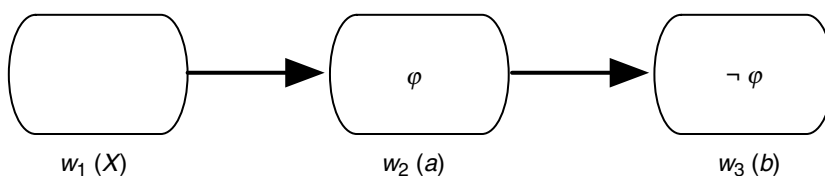
and if  $R$  had been transitive,  $w_3$  would have been accessible from  $w_1$ , and by definition of the semantics,  $\Box\varphi$  is **T** in a world iff  $\varphi$  is **T** in all worlds accessible from the world we are starting from (see definition 10.9).

With our approach, we could also have proved the *only if* part of the theorem (the *if* part has already been proved):

If  $R$  is not transitive, we graft to node 11. the tree:



By considering the branch going through nodes 7, 9, 10, 11, 13, and by applying the same method as in example 10.16, we obtain the Kripke model (interpretation):



which verifies:

$$\mathcal{M}, w_1 \models \Box\varphi$$

and:

$$\mathcal{M}, w_1 \not\models \Box\Box\varphi.$$

□

EXERCISE 10.7.– Use the translation method and the method of semantic tableaux to discover (*sufficient*) properties of the accessibility relations of the modal frames in which the following formulas are valid:

- a)  $\Box\varphi \Rightarrow \Diamond\varphi$
- b)  $\Diamond\varphi \Rightarrow \Box\Diamond\varphi$
- c)  $\Diamond\varphi \Rightarrow \Box\varphi$
- d)  $\Box(\Box\varphi \Rightarrow \varphi)$
- e)  $\Diamond\Box\varphi \Rightarrow \Box\Diamond\varphi$  (called *Geach's axioms*)

□

A question that naturally arises is whether other axioms (formulas) correspond to other simple relations, or if we can always express these relations by FOL formulas.

REMARK 10.10.– The problem of knowing whether an arbitrary wff of modal logic is satisfied by Kripke frames that are characterized by FOL formulas is *undecidable*. But Sahlqvist's theorem guarantees that a (syntactically characterized) class of formulas of modal logic corresponds to Kripke frames that are definable by FOL formulas. *Sahlqvist's class is decidable*.

The two following formulas do not belong to Sahlqvist's class.

*McKinsey's axiom:*

$$(MK) \quad \Box \Diamond \varphi \Rightarrow \Diamond \Box \varphi$$

$$(S4.1 = K \cup \{MK\})$$

and *Löb's axiom:*

$$\Box(\Box \varphi \Rightarrow \varphi) \Rightarrow \Box \varphi$$

cannot be defined by a set of wffs of FOL.

Furthermore, there exist formulas of modal logic that are valid in frames that are characterizable by FOL formulas, but are not equivalent to formulas in Sahlqvist's class, for example:

$$(\Box \Diamond \varphi \Rightarrow \Diamond \Box \varphi) \wedge (\Diamond \Diamond \varphi \Rightarrow \Diamond \varphi)$$

As a consequence, we can consider the expressive power of the PLs.

*Propositional modal logic corresponds to a fragment of SOL.* □

As can be seen, for example, in exercise 10.6, Aristotle was interested in other arguments than those of his syllogistic (see section 2.2). A well-known argument that he proposed on the need to philosophize is part of *deontic logic*:

We must philosophise or we must not philosophise. If we must philosophise then we must do so. Otherwise, we must still philosophise (to justify this point of view). Therefore, in all cases, we must philosophise.

EXERCISE 10.8.– How would you define the semantics of the following operators from *deontic logic*, using the semantics of possible worlds?

$O\varphi$ :  $\varphi$  is compulsory

$F\varphi$ :  $\varphi$  is forbidden

$P\varphi$ :  $\varphi$  is allowed

$E\varphi$ :  $\varphi$  is optional □



**10.4. Some elements of temporal logic**

When I am not asked, I know what time is; when I am asked, I no longer do.

*Saint Augustin*

The notion of time is essential in philosophy, in physics, in biology, and also in computer science (sequential and parallel algorithms, time complexity of algorithms, etc.).

In philosophy, it leads to different positions, it suffices to consider Parmenides for whom “nothing changes” and Heraclitus for whom “everything changes”. See also exercise 10.6. This notion is closely related to notions of space, matter, energy, evolution, etc. Its conceptualization poses huge problems and it is difficult to try avoiding them, as time is implicitly or explicitly present in all human activities (and all around us).

We often use metaphors to mention time (*the flow of time, time’s arrow, etc.*).

Similar to other topics, a logical study of time only aims at capturing *some* aspects that are relevant for the class of problems that are studied.

The notion of organization of instants:



seems to be the most fundamental one.

In mathematics (in the statement of definitions, theorems, etc.), we are not interested in time. This is not the case in natural languages. Linguists have studied the way time can be taken into account in logical structures.

For example (as is also common in physics):  $v_{t_0}, v_{t_1}, v_{t_2}, \dots$  (i.e. the speed of an object at times  $t_0, t_1, t_2, \dots$ ) or by explicitly introducing time as a parameter:  $interests(x, t), alive(x, t)$ , etc.

We can also (and this possibility is of a particular interest to us) treat instants as possible worlds (*now* corresponds to the world we are currently in). The accessibility relation will be a total order (linear time) or a partial one (branching time<sup>13</sup>).

Reference to the past and to the future depends on languages and is not always that obvious. For example, in English, we say:

---

<sup>13</sup> In branching time, we can have  $t_1 R t_2 \wedge t_1 R t_3 \wedge t_2 \neq t_3 \wedge \neg(t_2 R t_3 \vee t_3 R t_2)$ .

– the plane is scheduled to land at 6:30 pm (and it is currently 3:10 pm, we use the *present* tense to talk of the *future*);

– I now understand the rules of rugby (we are using the *present* tense to talk of an event that belongs in a large part to the *past*: the understanding of the rules).

Problems about the nature of time and temporal concepts have been a concern for philosophers at least since there has been a written philosophy.

Time is part of our sensory experience (at least indirectly), and our discourse relies heavily on time (in the sentences above, for example).

Classical mathematical logic abstracts time away. The propositions that are obtained from sentences from everyday life are sometimes very artificial.

The goal of temporal logic (also called *tense* or *change logic*) is to systematize reasoning with propositions involving temporal aspects.

The Megarian and Stoic Greeks studied theories for these logics.

Modern temporal logic was initiated by the work of logician and philosopher Arthur N. Prior in the 1950s. Prior systematically studied the links between temporal and modal logics.

The representation (picture) of time as *dots* that denote *instants* without duration as basic entities is frequently used. Examples are graphs representing functions of time (speed, distance travelled, GDP, unemployment rate, etc.) that are used to teach physics, economy, etc. where in each instant (point) on the abscissa is assigned an ordinate point.

We must distinguish between the graphs of functions that correspond to *continuous* phenomena and those that correspond to *discrete* phenomena.

Although common, the concept of time as a set of instants without duration is a very abstract model, compared to our perception in everyday life, and it is similar to the notion of a *point in space* (what exactly is a region of space, say, a sphere of radius 0?).

Another basic entity that is widely used is the *period* or *interval*.

Depending on the problem under consideration, one or the other notion will be better adapted.

In particular, there are problems that cannot be solved with one of these notions, but can be with the other (or not even be problems at all for the other). Here is an

example dating at least from the Middle Ages: *at the precise moment a man dies, is he alive or dead?*, or *at the precise moment a fire is put out, is it burning or extinct?*

This problem cannot be solved with the notion of instants, but with the notion of time interval, there is no problem at all: the man died between 2 and 3 am, if he was alive at 2 am and dead at 3 am.

If we use a discrete model for time, then there is no problem either.

Under some circumstances, it is more interesting to consider both notions at the same time. For example, if we say *Alice and Bob took the 6:42 train and chatted throughout the entire trip*, we are likely to be considering the departure time as an instant and the duration of the trip as an interval.

When the basic entities are instants, the relations *before* and *after* are used.

For periods or intervals, the following relations (among others) are used: *overlap*, *non-overlap*, *meet*, *start and finish*

We provide a graphical representation and logical translation of each relation, using as primitive relations the precedence relation ( $\prec$ ) and the interval inclusion relation ( $\sqsubseteq$ ). Given two intervals  $A$  and  $B$ , they can be related the following ways (among others):

**.Overlap:**



$$\exists x(x \sqsubseteq A \wedge x \sqsubseteq B)$$

(Note that  $x$  denotes an *interval*.)

**.Non-overlap:**



$$\forall x(x \not\sqsubseteq A \vee x \not\sqsubseteq B)$$

**.Meet**



$$(A \prec B) \wedge \neg \exists x(A \prec x \prec B)$$

**.Start**A B  ...

$$\neg\exists x((x \sqsubseteq A \wedge x \prec B) \vee (x \sqsubseteq B \wedge x \prec A))$$

**.Finish**A ... B 

$$\neg\exists x((x \sqsubseteq A \wedge B \prec x) \vee (x \sqsubseteq B \wedge A \prec x))$$

**10.4.1. Temporal operators and semantics**

As was mentioned in section 10.4, Prior introduced modern temporal logic essentially for philosophical purposes and he naturally focussed on linguistic constructions.

Natural languages and the inferences that are performed on their sentences provide examples of the peculiarities that come up when time is taken into account.

For example, from *The weather is nice* and *The weather is cold*, we can (correctly) deduce that *The weather is nice and cold*. From *The weather will be nice* or *The weather will be cold* we can deduce *The weather will be nice or cold*. But starting with *The weather will be nice* and *The weather will be cold*, we cannot (correctly) deduce that *The weather will be nice and cold*.

We introduce the following operators:

**F** $\varphi$ : *at some point in time, it will be the case that  $\varphi$  (it will be the case at least once)*;

**P** $\varphi$ : *it was the case at some point in time that  $\varphi$  (it was the case at least once)*;

**G** $\varphi$  :<sup>def</sup>  $\neg\mathbf{F}\neg\varphi$ : *it will always be the case that  $\varphi$* ;

**H** $\varphi$  :<sup>def</sup>  $\neg\mathbf{P}\neg\varphi$ :  *$\varphi$  was always the case in the past*.

For the Megarians (philosophers of the school of philosophy founded by Euclid of Megara)<sup>14</sup>:

---

<sup>14</sup> The author of the famous *Elements* was Euclid of Alexandria.

- the *real* is what is realized (i.e. true) now;
- the *possible* is what is realized (i.e. true) at some arbitrary time;
- the *necessary* is what is true all the time.

With modern notations, we have the following respective translations for “possible” and “necessary”:

$$\diamond\varphi :^{def} \varphi \vee \mathbf{F}\varphi \vee \mathbf{P}\varphi$$

$$\square\varphi :^{def} \varphi \wedge \mathbf{G}\varphi \wedge \mathbf{H}\varphi$$

For Stoicians:

- the *real* is what is realized (i.e. true) now;
- the *possible* is what is realized (i.e. true) now or at a future time;

in other words:

- a proposition is *possible* if it is true or will be true;
- the *necessary* is what is realized (i.e. true) or will be realized at all future times.

With modern notations, we have the following respective translations for “possible” and “necessary”:

$$\diamond\varphi :^{def} \varphi \vee \mathbf{F}\varphi$$

$$\square\varphi :^{def} \varphi \wedge \mathbf{G}\varphi$$

Note that with one or the other definition, “possible” and “necessary” keep their usual relationship.

#### 10.4.1.1. A famous argument

Some Greek philosophers credit the Greek philosopher of the Megarian school Diodorus Cronus (– 4th) with a famous argument (that was analyzed by Prior) called the *Master argument*, that seems to be the following:

- D1) Every true proposition about the past is necessary;
- D2) An impossible proposition cannot be a consequence of a possible proposition;
- D3) There exists a proposition that is possible, but is not and will not be true.

Diodorus concludes (the reasoning to reach the conclusion is unknown) that the set  $\{D1, D2, D3\}$  is unsatisfiable, which allowed him to characterize:

- the possible as what is or will be true;
  - the impossible as what, being already false, will not be true;
  - the necessary as what, being true, will not be false;
- (Some authors translate: the necessary as what *is* and *will be* always the case (true).);
- the unnecessary as what is already false, or will be false.

Logicians and modern philosophers tried to reconstruct and formalize Diodorus's reasoning to conclude that  $\{D1, D2, D3\}$  is unsatisfiable.

One of the formalizations translates Diodorus's assertions as follows:

$$D1') Pq \Rightarrow \Box Pq$$

$$D2') (\Box(p \Rightarrow q) \wedge \Diamond p) \Rightarrow \Diamond q$$

$$D3') \exists r(\Diamond r \wedge \neg r \wedge \neg Fr)$$

and adds the following two assertions that were (according to historians) accepted by the Greeks.

$$D4) (p \wedge Gp) \Rightarrow PGp$$

$$D5) \Box(p \Rightarrow HFr)$$

The proof of the unsatisfiability of  $\{D1', D2', D3', D4, D5\}$ :

$$1) \Diamond r \wedge \neg r \wedge \neg Fr \quad (D3')$$

$$2) \Diamond r \quad (1)$$

$$3) \Box(r \Rightarrow HFr) \quad (D5)$$

$$4) \Diamond HFr \quad (D2'), (2), (3)$$

$$5) \neg r \wedge G\neg r \quad (1)$$

$$6) PG\neg r \quad (5), (D4)$$

$$7) \Box PG\neg r \quad (6), (D1')$$

$$8) \neg \Diamond \neg PG\neg r$$

$$9) \neg \Diamond HFr$$

10)□ (4), (9)

The logic that corresponds to *Diodorean logic* is:

$$S4.2 = S4 \cup \{\diamond\Box P \rightarrow \Box\diamond P\}$$

meaning that we add Geach's axiom to *S4* (see exercise 10.7), which is valid in *directed* frames (directed relations correspond to the so-called (*strongly*) *confluent* in  $\lambda$ -calculus):

$$\forall w\forall v\forall x(wRv \wedge wRx \Rightarrow \exists u(vRu \wedge xRu))$$

For temporal frames  $(T, \leq)$ , it is often expressed as:

$$\forall x\forall y(x \in T \wedge y \in T \Rightarrow \exists v \in T(x \leq v \wedge y \leq v))$$

#### 10.4.2. A temporal logic

Given the language of PL enriched with operators  $F, P, G, H$  and formally defined by the following syntactic rules:

$$\varphi ::= P_i | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \varphi_1 \Rightarrow \varphi_2 | \varphi_1 \Leftrightarrow \varphi_2 | \diamond\varphi | F\varphi | P\varphi | G\varphi | H\varphi$$

$$\Box\varphi : \text{def } \neg\diamond\neg\varphi$$

a model for this language is a triple  $\mathcal{M} = (T, \prec, v)$

where:

$T$  is a set of instants.

We restrict ourselves as follows:

$\prec$  is a total order relation (see definition 3.25) % Other hypotheses (density, etc.) lead to other temporal logics

$$v \text{ (valuation) } v : \mathcal{P} \rightarrow 2^T$$

(where  $\mathcal{P}$  is the set of basic symbols and  $2^T$  is the set of subsets of  $T$ )

We will say that  $\varphi$  is true at instant  $t$  in model  $\mathcal{M}$ , which will be denoted by  $\mathcal{M}, t \models \varphi$  and by  $\mathcal{M} \models \varphi[t]$ , where relation  $\models$  is defined inductively as follows:

$$\mathcal{M} \models P[t] \text{ iff } t \in v(P) \text{ \% } P : \text{ propositional symbol}$$

$\mathcal{M} \models \neg\varphi[t]$  (also written  $\mathcal{M} \not\models \varphi[t]$ ) iff  $\text{not}(\mathcal{M} \models \varphi[t])$

$\mathcal{M} \models \varphi[t] \vee \psi[t]$  iff  $\mathcal{M} \models \varphi[t]$  or  $\mathcal{M} \models \psi[t]$

$\mathcal{M} \models \varphi[t] \wedge \psi[t]$  iff  $\mathcal{M} \models \varphi[t]$  and  $\mathcal{M} \models \psi[t]$

$\mathcal{M} \models \varphi[t] \Rightarrow \psi[t]$  iff  $\text{not}(\mathcal{M} \models \varphi[t])$  or  $\mathcal{M} \models \psi[t]$

$\mathcal{M} \models \varphi[t] \Leftrightarrow \psi[t]$  iff  $\mathcal{M} \models \varphi[t] \Rightarrow \psi[t]$  and  $\mathcal{M} \models \psi[t] \Rightarrow \varphi[t]$

$\mathcal{M} \models \mathbf{F}\varphi[t]$  iff there exists  $t' > t$   $\mathcal{M} \models \varphi[t']$

$\mathcal{M} \models \mathbf{P}\varphi[t]$  iff there exists  $t' < t$   $\mathcal{M} \models \varphi[t']$

$\mathcal{M} \models \mathbf{G}\varphi[t]$  iff for all  $t' > t$   $\mathcal{M} \models \varphi[t']$

$\mathcal{M} \models \mathbf{H}\varphi[t]$  iff for all  $t' < t$   $\mathcal{M} \models \varphi[t']$

A wff  $\varphi[t]$  is **T** iff  $(T, <, v) \models \varphi[t]$  for every valuation  $v$ ;  $\varphi$  is **T**, noted  $\models \varphi$ , iff  $\varphi[t]$  is **T** for all  $t \in T$ .  $\square$

#### 10.4.3. How to reason with temporal logics?

Similar to modal logics, we can envisage formal systems or translation methods. For the temporal logic defined in section 10.4.2, we will propose a *direct* method: the method of semantic tableaux. We apply the same process as for classical and many-valued logics. We shall start by the rules that define the semantics to define the syntactic rules (that respect this semantics) that will allow us to construct the tableau.

As in its previous versions, the method of semantic tableaux enumerates the sets of (partial) models of a set of temporal formulas, in which time occurs. The considerations of section 3.2 about the usage of the method apply.

The following remarks are important:

– there is a new parameter: time. As mentioned, we restrict ourselves to totally ordered instants:  $\dots, t_{-n}, t_{-(n-1)}, \dots, t_{-1}, t_0, t_1, t_2, \dots, t_n, \dots$ ;

– the contradictions that permit us to close a branch (and thus to conclude that it is impossible to construct the potential model that is represented by the branch) between a formula  $\varphi$  and its negation  $\neg\varphi$  must correspond to the same instant (which is rather obvious: there is no contradiction in the fact of being alive at instant  $t_i$  and not alive at instant  $t_j$ , where  $i \neq j$ !);

– when applied to a formula  $\varphi$  at instant  $t_i$ , the rules that are concerned with connectives  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\Rightarrow$ , and  $\Leftrightarrow$  will permit us to replace  $\varphi$  by the formulas denoting all models of  $\varphi$  at the same instant  $t_i$ . As usual, we shall tick  $\varphi$  as already analyzed ( $\surd$ );



– rules  $F$ ,  $P$ ,  $\neg G$ , and  $\neg H$  below apply only once and will therefore be ticked after being used ( $\checkmark$ ). Rules  $\neg F$ ,  $\neg P$ ,  $G$ , and  $H$  below can be applied at any desired instant. We will not be interested in the problem of decidable subclasses.

The rules are therefore the following:

10.4.3.1. *The method of semantic tableaux*

$\mathbf{F} : F\varphi[t]$ $\downarrow$ $\varphi[t'] \text{ for a } t' (t \prec t')$	$\neg\mathbf{F} : \neg F\varphi[t]$ $\downarrow$ $\neg\varphi[t'] \text{ for all } t' (t \prec t')$
$\mathbf{P} : P\varphi[t]$ $\downarrow$ $\varphi[t'] \text{ for a } t' (t' \prec t)$	$\neg\mathbf{P} : \neg P\varphi[t]$ $\downarrow$ $\neg\varphi[t'] \text{ for all } t' (t' \prec t)$
$\mathbf{G} : G\varphi[t]$ $\downarrow$ $\varphi[t'] \text{ for all } t' (t \prec t')$	$\neg\mathbf{G} : \neg G\varphi[t]$ $\downarrow$ $\neg\varphi[t'] \text{ for a } t' (t \prec t')$
$\mathbf{H} : H\varphi[t]$ $\downarrow$ $\varphi[t'] \text{ for all } t' (t' \prec t)$	$\neg\mathbf{H} : \neg H\varphi[t]$ $\downarrow$ $\neg\varphi[t'] \text{ for a } t' (t' \prec t)$

EXAMPLE 10.18.– Prove the validity (or non-validity) of the formula below, using the method of semantic tableaux:

$$F\varphi \Rightarrow F F\varphi$$

$\neg(F\varphi \Rightarrow F F\varphi)$	$\checkmark$	$t_0$
$\downarrow$		
$F\varphi$	$\checkmark$	$t_0$
$\neg(F F\varphi)$		$t_0$
$\downarrow$		
$\varphi$		$t_1$
$\neg F\varphi$		$t_1$
$\downarrow$		
$\neg\varphi$		$t_2$
$\downarrow$		
$\neg\varphi$		$t_3$
$\vdots$		

Of course, every  $t_i$  is different from the other  $t_j$ 's ( $i \neq j$ ).

The considered formula is not valid. Counter example (i.e. model of its negation):  $\{\varphi[t_1], \neg\varphi[t_2]\}$ . □

EXERCISE 10.9.– Use the method of semantic tableaux to prove the validity (or non-validity) of the formulas below:

a)  $F(\varphi \vee \psi) \Leftrightarrow F\varphi \vee F\psi$

b)  $F\varphi \wedge F\psi \Rightarrow F(\varphi \wedge \psi)$  □

EXERCISE 10.10.– Prove the validity of formula:

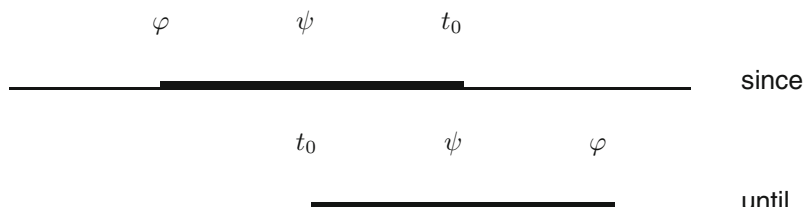
$$G((p \Rightarrow q) \Rightarrow p) \Rightarrow Gp$$

using the method of semantic tableaux. □

The two following operators are particularly important in computer science and confer a great expressive power to temporal logic. These are: **since** and **until**:

$S\varphi\psi$ : it was the case that  $\varphi$  and since then and up till now ( $t_0$ ) it was the case that  $\psi$ .

$U\varphi\psi$ : it will be the case that  $\varphi$  and from now ( $t_0$ ) until then it will be the case that  $\psi$ .



Their translation in FOL is:

$$S\varphi\psi : \exists t < t_0 (\varphi(t) \wedge \forall t' > t (t' < t_0 \Rightarrow \psi(t')))$$

$$U\varphi\psi : \exists t > t_0 (\varphi(t) \wedge \forall t' < t (t' > t_0 \Rightarrow \psi(t')))$$

In computer science, temporal logics are used in program verification, AI, databases, etc. They are used for the analysis and proofs of properties verified by many systems, including those that are concurrent, non-deterministic, real time, that do not end, etc.

The following relations are often studied in temporal logic:

irreflexivity:  $\forall x(x \not\prec x)$

asymmetry:  $\forall x\forall y(x < y \Rightarrow y \not\prec x)$

transitivity:  $\forall x\forall y\forall z(x < y \wedge y < z \Rightarrow x < z)$

there exists a beginning:  $\exists d\forall x(x \neq d \Rightarrow d < x)$

there exists an end:  $\exists f\forall x(x \neq f \Rightarrow x < f)$

linearity:  $\forall x\forall y(x = y \vee x < y \vee y < x)$

discrete:  $\forall x\forall y(x < y \Rightarrow \exists z(x < z \wedge \neg\exists u(x < u < z))) \wedge \forall x\forall y(x < y \Rightarrow \exists z(z < y \wedge \neg\exists u(z < u < y)))$

(To be compared with the notion of density below.)

density:  $\forall x\forall y(x < y \Rightarrow \exists z(x < z < y))$

Warning: density  $\neq$  continuity

DIGRESSION 10.2.– (on continuity). A total ordering  $(S, R)$  is *continuous* if it does not contain any *gap*.

A *cut* is a partition  $(X, Y)$  (i.e.  $S \subseteq X \cup Y$  and  $X \cap Y = \emptyset$ ) such that:

$\forall x\forall y(x \in X \wedge y \in Y \Rightarrow xRy)$

A *gap* is a cut  $(X, Y)$  such that  $X$  has no last element and  $Y$  has no first element.

$(\mathbb{Q}, \leq)$  has gaps: take the cut  $X = \{x \mid x^2 \leq 2\}$ ,  $Y = \{x \mid x^2 \geq 2\}$  □

REMARK 10.11.– The formula  $\Box(Gp \Rightarrow PGp) \Rightarrow (Gp \Rightarrow Hp)$  is valid in all continuous frames. □

#### 10.4.4. An example of a PL for linear and discrete time: PTL (or PLTL)

##### 10.4.4.1. Syntax

$\varphi ::= P_i | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \varphi_1 \Rightarrow \varphi_2 | \varphi_1 \Leftrightarrow \varphi_2 | \circ\varphi | \Diamond\varphi | \varphi_1 U \varphi_2$

$\Box\varphi$ :**def**  $\neg\Diamond\neg\varphi$

◦ : next

◇ : eventually

□ : always

$U$  : until

#### 10.4.4.2. Semantics

$\mathcal{M} = (T, S, v)$

$T$ : a finite or denumerable set of states (instants)

$S$ : total function (successor function):  $T \rightarrow T$

$s_0$ : first element

(as usual, we shall note  $S^i(t)$ ):

$\overbrace{S(S(S \dots (t) \dots))}^i$

$\mathcal{M}, t \models P$  iff  $P \in v(t)$

$\mathcal{M}, t \models \neg\varphi$  iff not  $(\mathcal{M}, t \models \varphi)$

$\mathcal{M}, t \models \varphi_1 \wedge \varphi_2$  iff  $\mathcal{M}, t \models \varphi_1$  and  $\mathcal{M}, t \models \varphi_2$

$\mathcal{M}, t \models \varphi_1 \vee \varphi_2$  iff  $\mathcal{M}, t \models \varphi_1$  or  $\mathcal{M}, t \models \varphi_2$

$\mathcal{M}, t \models \varphi_1 \Rightarrow \varphi_2$  iff non  $(\mathcal{M}, t \models \varphi_1)$  or  $\mathcal{M}, t \models \varphi_2$

$\mathcal{M}, t \models \varphi_1 \Leftrightarrow \varphi_2$  iff  $\mathcal{M}, t \models \varphi_1 \Rightarrow \varphi_2$  and  $\mathcal{M}, t \models \varphi_2 \Rightarrow \varphi_1$

$\mathcal{M}, t \models \circ\varphi$  iff  $\mathcal{M}, S(t) \models \varphi$

$\mathcal{M}, t \models \diamond\varphi$  iff  $\exists i \geq 0$  ( $\mathcal{M}, S^i(t) \models \varphi$ )

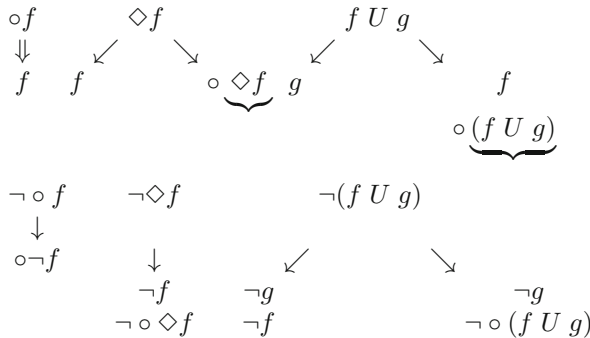
$\mathcal{M}, t \models \varphi_1 U \varphi_2$  iff  $\exists i \geq 0$  ( $\mathcal{M}, S^i(t) \models \varphi_2 \wedge \forall j (0 \leq j < i \Rightarrow \mathcal{M}, S^j(t) \models \varphi_1)$ )

An interpretation (model) satisfies a formula  $\varphi$  iff  $\mathcal{M}, s_0 \models \varphi$

10.4.4.3. Method of semantic tableaux for PLTL (direct method)

It is based on the same principle as the method used for PL and FOL, i.e. we enumerate all the models of a set of formulas. To do so, we propose the following rules that translate the semantics of the connectives that appear in the syntax (section 10.4.4.1). The rules that correspond to the classical connectives are the same as those for PL.

**Rules:**



$\rightarrow$ : decomposition of the wffs (independent of time). % see below;

$\Rightarrow$ : corresponds to moving forward one unit of time. % see below.

Once the principle mentioned above has been accepted, two new problems related to time arise:

- 1) the rules on  $\diamond$  and  $U$ ;
- 2) how to close a branch.

1) These rules can generate a potentially infinite tree, if we systematically choose the rightmost branches (they replace a formula by another formula containing the *same* connective).

To solve this problem, each time a formula is replaced, it is marked as already developed but still kept (we memorize that it has been developed). If at a given time (see below (\*)), say  $t_k$ , we obtain the same set of formulas as one occurring at a previous instant, say  $t_i$ , we add an edge:  $t_k \rightarrow t_i$ . This will lead to a directed graph instead of an infinite tree (see digression 3.1).

2) If at a given instant we have two contradictory literals  $P_i$  and  $\neg P_i$ , then the model we are trying to construct is not viable and we say the branch is *unsatisfiable*.

If there is a cycle in a branch, this means that an event  $(\diamond f, f U g)$  will *never* occur. We will therefore never be able to construct a model on the branch containing the cycle. The node from which the branch originates will be classified as *unsatisfiable*.

Of course, if all descendants of a node  $n$  are unsatisfiable, then  $n$  will be classified as *unsatisfiable*.

(\*) . . . But we still have not mentioned time!

Here is how we proceed: we first decompose (i.e. we apply the rules on) the formulas that do not begin with  $\circ$ . When they have all been decomposed, we decompose those beginning with  $\circ$  (*we move forward one unit of time*).

EXERCISE 10.11.– Use the direct method for PLTL to prove the unsatisfiability of the following formula:

$$\Box P \wedge \diamond \neg P$$

You will need to propose a rule for  $\Box P$  (use the definition from section 10.4.4.1).  $\square$

EXAMPLE 10.19.–

- 1)  $\Box(P \Rightarrow \circ Q)$
- 2)  $\Box(P \Rightarrow \circ(\neg QUR))$
- 3)  $\Box \diamond P$

1) is satisfied by every interpretation such that every state in which  $P$  is true is followed by a state in which  $Q$  is true.

2) is satisfied by every interpretation such that if  $P$  is true in a state then starting from the next state,  $Q$  is always false until the first state in which  $R$  is true.

3) is satisfied by every interpretation in which  $P$  is true infinitely many times.  $\square$

EXERCISE 10.12.– Can you justify the assertions (1), (2), and (3) of example 10.19, using the translation method and the direct method of semantic tableaux?  $\square$

## Chapter 11

# Knowledge and Logic: Some Notions

In computer science and AI, people often talk of “knowledge”, “knowledge representation”, “efficient usage of knowledge”, etc.

We may wonder whether logic, which allowed us, for example, to verify whether a reasoning is correct, to obtain conclusions from premises, to reason with fuzzy concepts, to qualify formulas as necessary or possible and to deduce consequences, to take time into account in formulas, and to reason on these formulas... will also allow us to talk of (and define) knowledge and to reason while taking into account the knowledge of an agent in a world where there are other agents, possibly with a different knowledge.

The goal of this chapter is to try to answer this question.

Up to now, we have assumed (implicitly or explicitly) that “knowledge” was more or less a synonym of “premise” and that the initial premises did not evolve when they interacted with the environment.

But while handling problems in which the state of knowledge may change, new difficulties will arise.

In game theory (an entire discipline by itself that we only mention here), it is also necessary to take an environment that is not passive into account, meaning that the actions of the other agents in the environment are relevant in the design of strategies to reach a goal.

Recall that the ideas on probabilities play a very important role in knowledge theory (for the time being we admit the intuitive meaning of knowledge): it suffices

to think of natural science. We may define probability as a numerical encoding of a state of knowledge (a probability 1 would mean a certain event and a probability 0 an impossible event)<sup>1</sup>.

The notion of “information” is closely related to that of knowledge, and therefore to that of probability. Information can be defined as anything that leads to a change in the assignment of probabilities<sup>2</sup>.

From a historical point of view, it is interesting to note that until 1660, probabilities were *in opposition with* knowledge, and in Galileo’s times, for example, sets of experiments did not replace proofs<sup>3</sup>.

We may now move on to the next stage: specify the abstractions that will allow us to formalize things.

### 11.1. What is knowledge?

A few questions and remarks naturally arise.

– What interest is there (for computer science and AI among others) to study this concept and possibly those that are related?

– A little of introspection is sufficient to imagine that those problems related to knowledge must have been studied by philosophers for a very long time: this is indeed the case.

– The Greeks and medieval philosophers were interested in this problem but experts seem to agree on the fact that the topic was studied systematically starting with Descartes, Leibniz, Hume and others (17th Century). A *theory of knowledge* was only envisaged starting with Kant (18th Century).

– The notion of knowledge seems inseparable from that of environment, and implicitly from our senses that allow us to interact with it.

– Philosophers propose: “to know is the action during which a subject (in computer science or AI we would say an *agent*) comprehends an object”, thanks to a *representation* (today we would say thanks to a model). It is interesting to note that Stoicians already used the notion of representation.

– Of course, problems arise, such as: “is it possible to know the entire environment (all of reality), or will we only be able to comprehend part of it?” (Note the similarity

---

1 See definition 11.1.

2 This definition uses “knowledge” as a primitive concept. We shall give a formal definition of this concept in definition 11.1.

3 See section 8.4.



with the difficulties experienced during the act of *modeling*.) These problems have to be tackled by all those who study natural science, especially physicists.

- The notion of “reality” itself leads to some problems.
- Philosophers talk of “sensitive reality” and “intelligible reality”.
- Sensitive reality can often be misleading (think, for example, of optical illusions, the principle of inertia, the free fall of bodies of different weights, quantum mechanics, etc.).
- This characteristic is the reason why intelligible reality is often considered as superior to sensitive reality.
- Some schools of philosophy that considered the notion of “experience” as a key notion studied it from a broad point of view, i.e. not only as an account of phenomena but also taking into account an elaboration that can be intellectual, historical, introspective, etc.

At this stage, we can already try to identify the characteristics that we want to keep in our formalization.

- The social factor (several knowing *agents*) in knowledge and interaction between agents (agent *X* knows that agent *Y* does not know that agent *Z* ...). Furthermore, it is clear that science, for example, is a social production.
- Introspection (*X* knows that he knows, ...).
- Related to inner experience: what difference between *knowledge* and *belief*?
- About the differences between knowledge and belief, they are opposed by the current discourse, but we can say that there is some sort of belief in knowledge, for example, that the laws of the universe do not change (with time) and that it is possible to know them.
- Belief could be defined as an adherence to an idea, an opinion, that we acknowledge as true and, partly at least, as subjective.
- It is related to the notion of *probability*, of which one definition is the *degree of belief*.
- Etymology does not seem capable of helping us, this time.
- In the search for a logical formalization, we shall make use of the following definitions, among others, that are proposed by philosophers (even if they are difficult to grasp).
- Knowledge: act of thought that penetrates and defines the object of its knowledge.
- The perfect knowledge of a thing, in this sense, is that which, when considered subjectively, does not leave anything obscure or unclear in the known thing, or that which, when considered objectively, does not leave anything that exists in the reality in which it applies outside.

- Belief: in a weak and large sense, it is the equivalent of an opinion, and denotes an imperfect consent that, similarly to an opinion, can have any degree of probability.
- . . . and we call probability (absolutely speaking) the character of the event it is the most reasonable to expect.
- To synthesize, we shall retain an *environment* of which the *knowing agent(s)* is (are) a particular case.

Among the logics that we have studied, modal logics (see section 10.3) are those that seem most naturally suited to be logics of knowledge because of the concepts they handle; the following definition is therefore oriented toward using these logics.

DEFINITION 11.1.– (knowledge).

- An agent knows a fact  $\varphi$  if  $\varphi$  is true in all world it considers as possible given its current knowledge.
- An agent considers as possible the worlds that are not in contradiction with the facts it holds as indubitable.
- An agent considers a fact  $\varphi$  as possible if it does not know  $\neg\varphi$ .

REMARK 11.1.– (on the definition of knowledge).

- “Given its current knowledge” naturally leads to thinking that knowledge can change with time, which in turn naturally leads to think of the notion of learning.
- In general (always?) we have *partial information* about the environment.
- We could say that the more information in the possession of an agent, the less possible worlds it will consider. In other words, the number of worlds it considers as possible is directly related to its uncertainty (think of the scientific process or of a game of cards, etc.). For example, if an agent rolls a die, there are six possible worlds. If it makes an object fall, there is only one possible world.  $\square$

DIGRESSION 11.1.– The knowledge modeling that is used in game theory or in mathematical economics uses concepts that are close to probabilities, in which a key notion is that of an *event*, i.e. the set of possible worlds.

We could say that instead of working with logical formulas, those working in these disciplines work directly on their denotation: for example, “ $n$  is an odd number” will be replaced by the set of worlds in which  $n$  is odd. “ $n$  is odd and greater than 2007” will be replaced by the intersection of the set of all worlds (states) in which  $n$  is odd, and the set of worlds (states) in which  $n > 2007$ <sup>4</sup>.  $\square$

---

<sup>4</sup> In the initial developments of probability theory, people talked of “propositions” with the meaning of events. It is after Kolmogorov’s axiomatization that elementary events were treated as sets. Stochastic situations can be modeled indifferently using sets or propositions.

## 11.2. Knowledge and modal logic

### 11.2.1. Toward a formalization

The description language must be able to express the retained notions:

- We assume  $n$  agents ( $n \in \mathbb{N}$ ).
- Propositions on the states of the world (or on the worlds) (for example, *the weather is nice, I have an ace*, etc.).  $\mathcal{P}$ : the set of all atomic propositions (formulas).
- $K_i$ : agent  $i$  knows  $\varphi$ .

### 11.2.2. Syntax

$$\begin{aligned}
 - \varphi ::= & P_i | \neg\varphi | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \varphi_1 \Rightarrow \varphi_2 | \varphi_1 \Leftrightarrow \varphi_2 | K_i\varphi \\
 \mathbf{T} : \text{def} & P \vee \neg P \\
 \mathbf{F} : \text{def} & \neg \mathbf{T}
 \end{aligned}$$

#### 11.2.2.1. What expressive power? An example

- We will be able to write formulas such as (proposition  $P$  meaning: *agent 1 has an ace*)

$$K_1 K_2 P \wedge \neg K_2 K_3 P$$

- meaning: *agent 1 knows that agent 2 knows that he (agent 1) has an ace and agent 2 does not know that agent 3 knows that agent 1 has an ace.*

#### 11.2.2.2. Semantics

An e-frame (e stands for epistemic):

- $\mathcal{M} = (W, K_1, K_2, \dots, K_n, v)$
- $W \neq \emptyset$ : set of possible worlds
- $K_i \subseteq W^2$ : accessibility relations of the agents
- $v : \mathcal{P} \longrightarrow 2^W$  or  $W \longrightarrow 2^{\mathcal{P}}$  where, as usual,  $2^W$  and  $2^{\mathcal{P}}$  denote sets of subsets of  $W$  and  $\mathcal{P}$ , respectively. ( $v$  is called a *valuation* and indistinctly gives the set of worlds in which a proposition is evaluated to  $\mathbf{T}$  or the set of propositions that are evaluated to  $\mathbf{T}$  in each world).

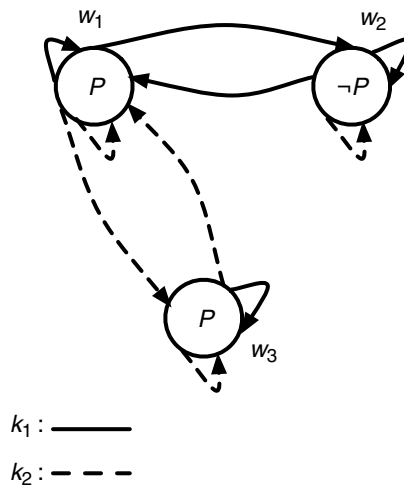
$$- \mathcal{M}, w \models K_i \varphi \text{ iff for all } w' \text{ such that } (w, w') \in K_i : \mathcal{M}, w' \models \varphi$$

We can see that, with the semantics of the accepted definition of knowledge, the latter is relative to the possible worlds (i.e. to the world we belong to) and to the agents (i.e. to the worlds that are accessible to them).

EXAMPLE 11.1.– (Fagin *et al.*).

- $\mathcal{M} = (W, K_1, K_2, v)$
- $W = \{w_1, w_2, w_3\}$
- $v(P) = \{w_1, w_3\}$
- $K_1 = \{(w_1, w_1), (w_1, w_2), (w_2, w_1), (w_2, w_2), (w_3, w_3)\}$
- $K_2 = \{(w_1, w_1), (w_1, w_3), (w_2, w_2), (w_3, w_1), (w_3, w_3)\}$

The accessibility relations and the valuation are, similarly to the other modal logics, represented in a compact way as a graph:



In world  $w_3$ , agent 1 knows that  $\mathcal{P}$ , but in world  $w_1$ , he does not. Agent 2 knows that  $\mathcal{P}$  in both worlds, but does not in world  $w_2$ . □

To summarize, operator  $K_i$  permits descriptions in which:

- A world (state) *is not* completely characterized by  $v$ . Each agent has its “particular view” of reality. For example, if  $P$  is an arbitrary proposition ( $\mathbf{T}$  in world  $w_1$ ), it is possible for agent 1 not to know  $P$ , and for agent 2 to know  $P$  but not to know that agent 1 does not know  $P$ , ...:

$$\mathcal{M}, w_1 \models P \wedge \neg K_1 P \wedge K_2 P \wedge \neg K_2 \neg K_1 P \wedge K_1 (K_2 P \vee K_2 \neg P) \quad \square$$

We will say that a formula  $\varphi$  is valid in an e-frame

$\mathcal{M} = (W, K_1, K_2, \dots, K_n, v)$ , noted  $\mathcal{M} \models \varphi$  iff for all  $w \in W : \mathcal{M}, w \models \varphi$ .

$\varphi$  is *satisfiable* in  $\mathcal{M}$  iff there exists  $w \in W$  such that  $\mathcal{M}, w \models \varphi$ .

Finally,  $\varphi$  is *valid* iff  $\varphi$  is valid in all e-frames, and  $\varphi$  is *satisfiable* iff it is satisfiable in an e-frame.

### 11.2.3. New modal operators

To formalize the abstractions that were retained for knowledge, three modal operators are introduced ( $G \neq \emptyset$ : subset of agents).

- $E_G$ : *all in  $G$  know*.
- $C_G$ : it is a *common knowledge* of all the agents in  $G$  (i.e. all know, all know that all know, ...).
- $D_G$ : it is a *distributed knowledge* among the agents in  $G$  (i.e. the agents in  $G$  know in the worlds they have access to, in other words,  $\varphi$  is a distributed knowledge among the agents in  $G$  iff  $\varphi$  is **T** in all worlds that are accessible by all the agents in  $G$ ).

If  $G$  is the set of *all* agents, we shall write  $E$ ,  $C$  and  $D$ .

We extend the syntax to take these new operators into account:

#### 11.2.3.1. Syntax (extension)

- If  $\varphi$  is a formula, then  $E_G\varphi$ ,  $C_G\varphi$ ,  $D_G\varphi$ ,  $E\varphi$ ,  $C\varphi$ ,  $D\varphi$  are also formulas. Thus, we must add to the grammar of section 11.2:
- $\varphi ::= E_G\varphi \mid C_G\varphi \mid D_G\varphi \mid E\varphi \mid C\varphi \mid D\varphi$

... and we must also formally define the semantics of these operators:

#### 11.2.3.2. Semantics (extension)

- $\mathcal{M}, w \models E_G\varphi$  iff for all  $i \in G$ :  $\mathcal{M}, w \models K_i\varphi$
- We note:
  - $E_G^0\varphi : \varphi$
  - $E_G^1\varphi : E_G\varphi$  % i.e. all those in  $G$  know  $\varphi$
  - $E_G^2\varphi : E_G E_G\varphi$  % i.e. all those in  $G$  know that all those in  $G$  know  $\varphi$
  - $E_G^{i+1}\varphi : E_G E_G^i\varphi$
  - $\mathcal{M}, w \models C_G\varphi$  iff  $\mathcal{M}, w \models E_G^i\varphi$  for  $i = 1, 2, \dots$
  - $\mathcal{M}, w \models D_G\varphi$  iff for all  $w'$  such that  $(w, w') \in \bigcap_{i \in G} K_i$ , we have  $\mathcal{M}, w' \models \varphi$
- For example we can write:
  - $K_1 \neg C_{\{2,3\}} P \wedge \neg C Q \wedge D Q$   
 meaning: *Agent 1 knows that  $P$  is not a common knowledge of agents 2 and 3 and  $Q$  is not a common knowledge, but a distributed knowledge in the system.*

### 11.2.4. Application examples

#### 11.2.4.1. Modeling *the muddy children puzzle*

The problem:

- $n$  children (brothers) are playing in a muddy courtyard. Their mother told them that those who got mud on themselves would be punished.
- During the game,  $k$  children get mud stains on their foreheads.
- Of course, each child can see the mud stains on the others' foreheads, but not on their own.
- Each child hides what he sees.
- Their father, who always tells the truth, comes home and says: *at least one of you have a mud stain on your forehead* (a fact which, if  $k > 1$  was known by all. If  $k = 1$ , then the one with the stain on the forehead did not know that).
- The father repeats again and again the question: “does one of you know that he has a mud stain on the forehead?”
- We assume that the children reason perfectly and that they never lie.
- Thus (this is what needs to be proved) during the first  $k - 1$  times the question is repeated, the children will answer “no”, but when the father asks the question for the  $k^{\text{th}}$  time, the children with mud stains on their forehead will answer “yes”.

The modeling:

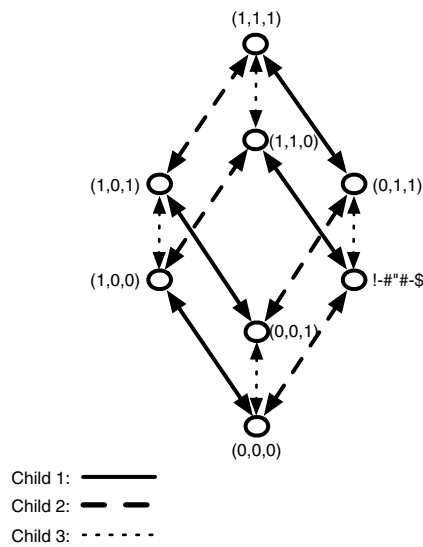
- Similarly to all modeling phases, the notions of “good” models and information are implicit. There is common knowledge that is implicit (for example, no child is blind).
- $P_i$ : child  $i$  has a mud stain on his forehead.
- Each world will be characterized (named) by an  $n$ -tuple  $(x_1, x_2, \dots, x_n)$ ;  $x_i \in \{0, 1\}$   
(with the meaning: “child  $i$  has a mud stain (has no mud stain) on the forehead iff  $x_i = 1$  ( $x_i = 0$ )”)
- Thus,  $\mathcal{M}, (x_1, x_2, \dots, x_n) \models P_i$  iff  $x_i = 1$ .
- We, therefore, define  $v$ , taking this into account.
- For the sake of simplicity we can define:  
 $P_i \stackrel{def}{=} P_1 \vee P_2 \vee \dots \vee P_n$

#### 11.2.4.2. Corresponding Kripke worlds

We assume there are three children. The graph below represents all possible configurations of mud stains on the children's foreheads ( $2^3$  possibilities). The triples  $(i, j, k)$  ( $i, j, k \in \{0, 1\}$ ) correspond to the valuations (here, we also use them as names of worlds or states). The (double) arrows denote the world (or state) that each

child considers as possible (i.e. w.r.t. the accessibility relation) in the world in which he is (i.e. where he sees what he sees).

For the sake of readability, we did not draw for each child and each world the arrows with the same origin and destination.



- We can write, for example:
- $\mathcal{M}, (1, 1, 0) \models K_3 P_2$
- $\mathcal{M}, (1, 0, 1) \models K_1 \neg P_2$
- $\mathcal{M}, (1, 0, 1) \models K_1 P_3$
- $\mathcal{M}, (1, 0, 1) \models \neg K_1 P_1$
- $\mathcal{M} \models C(P_2 \Rightarrow K_1 P_2)$
- $\mathcal{M} \models C(\neg P_2 \Rightarrow K_1 \neg P_2)$
- $\mathcal{M}, (1, 0, 1) \models EP$

The father speaks  $\rightsquigarrow$  the graph changes.

- For example, in  $(1, 0, 1)$ , child 1 considers  $(0, 0, 1)$  as possible, and in this world, 3 considers  $(0, 0, 0)$  as possible.

- Hence, 1 considers as possible that 3 considers as possible that no child has a mud stain on the forehead.

- When the father says  $P$ , things change:

- No child considers state  $(0, 0, 0)$  is possible anymore.

11.2.4.3. *Properties of the (formalization chosen for the) knowledge*

Similarly to all modeling phases, the abstraction adopted for knowledge must be subject to analysis to evaluate its pertinence. The researchers who proposed it note that the properties and consequences of the definition of knowledge that was accepted correspond to what we informally expect of this concept.

For example, we can prove:

$$- \models (K_i\varphi \wedge K_i(\varphi \Rightarrow \psi)) \Rightarrow K_i\psi \quad \% \text{ Distribution axiom};$$

$$- \text{ For all } \mathcal{M} \text{ if } \mathcal{M} \models \varphi \text{ then } \mathcal{M} \models K_i\varphi \quad \% \text{ Knowledge generalization rule}$$

This property states that an agent knows every *valid* formula (...but not necessarily those that are true).

$$- \models K_i\varphi \Rightarrow \varphi \quad \% \text{ Knowledge axiom}$$

knowledge  $\neq$  *belief* (see example 10.17)

This property states that an agent does not know something false.

$$- \models K_i\varphi \Rightarrow K_iK_i\varphi \quad \% \text{ Positive introspection axiom}$$

(see example 10.17)

$$- \models \neg K_i\varphi \Rightarrow K_i\neg K_i\varphi \quad \% \text{ Negative introspection axiom}$$



## Chapter 12

### Solutions to the Exercises

EXERCISE 3.1.— We can represent an interpretation as a subset of  $\Pi$ , with the elements of this subset being the propositional symbols that are evaluated to  $\mathbf{T}$  in the interpretation.

There are  $2^{\aleph_0}$  subsets of  $\Pi$ ; hence, there is an *uncountably infinite number of interpretations for PL*.

A more detailed proof.

Every interpretation  $I$  can be represented by the graph of the function defining it:

$$I = \{(P_j, *) \mid j \in \mathbb{N}, * : \mathbf{T} \text{ (exclusive) or } * : \mathbf{F}\}$$

There are infinitely many interpretations, take for example ( $l \in \mathbb{N}$ ):

$$I_l = \{(P_l, \mathbf{T}), (P_m, \mathbf{F}) \mid m \in \mathbb{N}; l \neq m\}$$

Now if we *assume* that the set of all interpretations in PL is *denumerable*, then they can be enumerated ( $i \in \mathbb{N}$ ):

$$I_i = \{(P_j, *) \mid j \in \mathbb{N}, * : \mathbf{T} \text{ (exclusive) or } * : \mathbf{F}\}$$

Consider the interpretation  $\mathcal{I}$  defined as follows:

$$\mathcal{I} = \{(P_k, *) \mid k \in \mathbb{N}, * : \mathbf{T} \text{ if } (P_k, \mathbf{F}) \in I_k; * : \mathbf{F} \text{ if } (P_k, \mathbf{T}) \in I_k\}$$

This interpretation is different from every  $I_i$  in the list we assumed we were able to construct, at least for the couple  $(P_i, *)$ .

Therefore, assuming that the set of interpretations for PL is denumerable leads to a *contradiction*. The set of all interpretations of PL is therefore uncountably infinite.  $\square$

REMARK 12.1.– To avoid the (implicit) use of the axiom of choice, we can encode  $F : 0 \ T : 1$  and define:

$$\mathcal{I} = \{(P_k, (* + 1)_{mod\ 2}) \mid k \in \mathbb{N}; (P_k, *) \in I_k, * = 0 \text{ or } * = 1\}. \quad \square$$

REMARK 12.2.– The argument used here is the same as the argument used to prove that there are uncountably many real numbers in the interval  $[0, 1]$  (which entails that  $\mathbb{R}$  is uncountably infinite):

We assume that we can enumerate all real numbers between 0 and 1:

$$\begin{array}{l} 0, x_1^1 x_2^1 \dots x_n^1 \dots \quad x_j^i \in [0, 1, 2, \dots, 9] (i, j = 1, 2, \dots) \\ 0, x_1^2 x_2^2 \dots x_n^2 \dots \\ \vdots \\ 0, x_1^p x_2^p \dots x_n^p \dots \\ \vdots \end{array}$$

But the real number:

$$y = 0, y_1 y_2 \dots y_1 \dots, \text{ where } y_i \neq x_i^i \ i = 1, 2, \dots$$

is not in this list (it is different from the first number at least at  $x_1^1$ , from the second at least at  $x_2^2$ , ... from the  $n^{\text{th}}$  at least at  $x_n^n$ , ...)

A contradiction; hence, the set of real numbers on  $[0, 1]$  is not denumerable.  $\square$

EXERCISE 3.2.–

a) A truth function can always be represented by its graph, a its domain and range are finite.

The following *example* clearly shows how to proceed (it is trivial to transform this method into an algorithm).

$P$	$Q$	$R$	$f(P, Q, R)$	$\neg f(P, Q, R)$
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>

The dnf of  $f(P, Q, R)$  is:

$$(P \wedge Q \wedge R) \vee (P \wedge Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge \neg R).$$

The justification is very simple (properties of  $\wedge$  and  $\vee$ ): every other combination of literals that appears in the dnf is equivalent to **F** (by inspection of the truth table).

The cnf of  $f(P, Q, R)$  is:

$$(\neg P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (P \vee \neg Q \vee \neg R) \wedge (P \vee \neg Q \vee R) \wedge (P \vee Q \vee \neg R).$$

It is obtained because of the following equivalences:

$$\neg \bigvee \bigwedge P_i \text{ equiv. } \bigwedge \bigvee \neg P_i$$

and:

$$\neg \neg f(P, Q, R) \text{ equiv. } f(P, Q, R)$$

i.e. we take the conjunction of disjunctions for those  $f(P, Q, R) = \mathbf{F}$ .

b) Use the equivalence  $P \vee Q \text{ equiv. } \neg(\neg P \wedge \neg Q)$  and (a).

c) Use the equivalence  $P \wedge Q \text{ equiv. } \neg(\neg P \vee \neg Q)$  and (a).

d) Use the equivalences  $P \wedge Q \text{ equiv. } \neg(P \Rightarrow \neg Q)$   $P \vee Q \text{ equiv. } \neg P \Rightarrow Q$  and (a).

e) Use the equivalences  $\neg P \text{ equiv. } P \mid P$ ;  $P \vee Q \text{ equiv. } ((P \mid P) \mid (Q \mid Q))$  and (c).

f) Use the equivalences  $\neg P \text{ equiv. } P \downarrow P$ ;  $P \wedge Q \text{ equiv. } ((P \downarrow P) \downarrow (Q \downarrow Q))$  and (b).  $\square$

EXERCISE 3.3.– We provide a sufficient condition.

It is true if the only line with value **F** is the given one (see **cnf**, solution of exercise 3.2) or if the only line with value **T** is the one given (see **dnf**, solution of exercise 3.2).  $\square$

EXERCISE 3.4.–

a)

$P$	$\wedge$	$(P \Rightarrow Q)$	$\Rightarrow$	$Q$
<b>T</b>	<b>T</b>	<b>T T T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>T F F</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>F T T</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>F T F</b>	<b>T</b>	<b>F</b>

b)

$(P \Rightarrow Q)$	$\wedge$	$(\neg Q \Rightarrow \neg P)$
<b>T T T</b>	<b>T</b>	<b>F T F</b>
<b>T F F</b>	<b>F</b>	<b>T F F</b>
<b>F T T</b>	<b>T</b>	<b>F T T</b>
<b>F T F</b>	<b>T</b>	<b>T T T</b>

c)

$\neg A$	$\wedge$	$(A \vee B)$	$\Rightarrow$	$B$
<b>F</b>	<b>F</b>	<b>T T T</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>T T F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F T T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F F F</b>	<b>T</b>	<b>F</b>

d)

$A$	$\Rightarrow$	$A \vee B$
<b>T</b>	<b>T</b>	<b>T T T</b>
<b>T</b>	<b>T</b>	<b>T T F</b>
<b>F</b>	<b>T</b>	<b>F T T</b>
<b>F</b>	<b>T</b>	<b>F F F</b>

e)

$(P \Rightarrow Q)$	$\wedge$	$\neg Q$	$\Rightarrow$	$\neg P$
<b>T T T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>T F F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>F T T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>
<b>F T F</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>

□

EXERCISE 3.5.-

a)

To know whether it is always **F**, we try to evaluate it to **T**:

to evaluate to **T** an implication, it suffices (this is not the only possibility) to fix its consequent to **T**.

$$(\dots) \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{\left( \overset{\mathbf{F}}{S} \Rightarrow (R \vee T) \right)}_{\mathbf{T}}$$

It thus suffices to assign **F** to  $S$  and not bother with the truth values of the other propositional symbols to evaluate the given formula to **T**. It is not always **F**.

To know whether it is always **T**, we try to evaluate it to **F**:

to evaluate an implication to **F**, the only possibility is to have its first term evaluate to **T** and its second term evaluate to **F**.

$$((\dots \Rightarrow \underbrace{Q}_{\mathbf{T}}) \wedge (\dots \Rightarrow \underbrace{\neg R}_{\mathbf{T}})) \wedge (\underbrace{R}_{\mathbf{F}} \Rightarrow \dots) \wedge (\dots \Rightarrow \underbrace{Q}_{\mathbf{T}}) \stackrel{\mathbf{F}}{\Rightarrow} (\underbrace{S}_{\mathbf{T}} \Rightarrow (\underbrace{R}_{\mathbf{F}} \vee \underbrace{T}_{\mathbf{F}}))$$

We can evaluate it to **F** with the given assignments . It is not always **T**.

*Conclusion:* the given formula is neither contradictory nor a tautology.

b)

As mentioned above, to evaluate an implication to **F**, the only possibility is to evaluate its first term to **T** and its second term to **F**.

To evaluate the second term to **F**, the only possibility is:

$$\underbrace{(E \Rightarrow A)}_{\mathbf{T}} \stackrel{\mathbf{F}}{\Rightarrow} \underbrace{(D \Rightarrow A)}_{\mathbf{F}}$$

and the only possible assignments are:

A: **F**

D: **T**

hence:

E: **F**

With these assignments, we try to evaluate the first term to **T**, the only possibility is:

$$\underbrace{(((A \Rightarrow B) \Rightarrow (\neg C \Rightarrow \neg D)) \Rightarrow C)}_{\mathbf{F}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{E}_{\mathbf{F}}$$

With the only possible assignments for A, D, and E, we must assign C to **T** and **F**, which is impossible.

*Conclusion:* the given formula cannot be evaluated to **F**. □

EXERCISE 3.6.– No,  $\preceq$  is not anti-symmetric (see definition 3.23):

Of course, for example:

$$F \models F \vee F \text{ and } F \vee F \models F$$

But formula  $F$  is (syntactically) different from formula  $F \vee F$ . □

EXERCISE 3.7.–

a)

$P$	$\Rightarrow$	$(Q \Rightarrow P)$
<b>T</b>	<b>T</b>	<b>T T T</b>
<b>F</b>	<b>T</b>	<b>T F F</b>
<b>T</b>	<b>T</b>	<b>F T T</b>
<b>F</b>	<b>T</b>	<b>F T F</b>

*If we have  $P$ , any proposition implies  $P$ .*

b)

$\neg P$	$\Rightarrow$	$P \Rightarrow Q$
<b>F</b>	<b>T</b>	<b>T T T</b>
<b>F</b>	<b>T</b>	<b>T F F</b>
<b>T</b>	<b>T</b>	<b>F T T</b>
<b>T</b>	<b>T</b>	<b>F T F</b>

*If  $P$  is false, then  $P$  implies any proposition.*

(a) and (b) are called *paradoxes of material implication*. □

EXERCISE 3.8.– As is customary (in particular in mathematics), we convene that a reasoning is correct if and only if

We cannot evaluate the premises to **T** and the conclusion to **F**. □

REMARK 12.3.– This convention (definition) enables us to immediately propose two classes of reasonings that are trivially correct:

contradictory premises (inclusive) or tautological conclusion. □

Before any translation, it is necessary to identify the synonyms and propositions that are negations of other propositions.

We propose:

$S$ : Life has a meaning.

$M$ : Life necessarily ends by death.

$T$ : Life is sad.

$C$ : Life is a cosmic joke.

$A$ : Angst exists.

$B$ : Life is beautiful.

$\neg B$ : Life is ugly (life is not beautiful).

$\neg M$ : Life goes on after death.

Most of the time we must make translation choices, but we will assume (this is not important, the important fact is to be aware of the difficulty of translation) that we all agree with the translation.

Once the translation has been made, we try (*without forgetting to test the other possibilities if there is a failure*) to find an interpretation that permits us to evaluate the conclusion to **F** and the premises to **T**.

If we succeed, the reasoning is incorrect and we have produced a counter example.

If we fail, the reasoning is correct.

$$\begin{array}{c}
 \underbrace{S}_{\mathbf{F}} \wedge \underbrace{M}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{T}_{\mathbf{F}} \\
 \underbrace{\neg M}_{\mathbf{F}} \wedge \underbrace{\neg S}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{C}_{\mathbf{F}} \\
 \underbrace{S}_{\mathbf{F}} \wedge \underbrace{\neg M}_{\mathbf{F}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{\neg A}_{\mathbf{F}} \\
 \underbrace{\neg C}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{\neg T}_{\mathbf{T}} \\
 \underbrace{A}_{\mathbf{T}} \\
 \underbrace{T}_{\mathbf{F}} \vee \underbrace{C}_{\mathbf{F}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{\neg B}_{\mathbf{F}} \\
 \hline
 \underbrace{\neg B}_{\mathbf{F}}
 \end{array}$$

Therefore, with this formalization, the reasoning is incorrect.  $\square$

## EXERCISE 3.9.–

a)

$$\begin{array}{c}
 \underbrace{A}_{\mathbf{F}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{B}_{\mathbf{F}} \\
 \underbrace{A}_{\mathbf{F}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{C}_{\mathbf{F}} \\
 \neg(\underbrace{B}_{\mathbf{F}} \vee \underbrace{C}_{\mathbf{F}}) \\
 \hline
 \underbrace{D}_{\mathbf{F}}
 \end{array}$$

Incorrect reasoning. Which is “natural”: *in general*, when the conclusion is independent from the premises, the reasoning is incorrect, except in the case of reasonings that are trivially correct (i.e. contradictory premises and/or tautological conclusion).

b)

$$\begin{array}{c}
 \underbrace{A}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{B}_{\mathbf{T}} \\
 \underbrace{B}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{C}_{\mathbf{T}} \\
 \underbrace{C}_{\mathbf{T}} \stackrel{\mathbf{T}}{\Rightarrow} \underbrace{D}_{\mathbf{T}} \\
 \neg D \quad \quad \quad \% \text{ impossible to evaluate to } \mathbf{T} \\
 \underbrace{A}_{\mathbf{T}} \vee \underbrace{E}_{\mathbf{F}} \\
 \hline
 \underbrace{E}_{\mathbf{F}}
 \end{array}$$

As there is no other possible way of evaluating the conclusion to  $\mathbf{F}$  and the premises to  $\mathbf{T}$ , we conclude that the reasoning is correct.

If we had wanted to go from the premises to the conclusion (*forward chaining*), we would have done:



$\neg D \mapsto \mathbf{T}$  (only possibility), hence (contrapositive)  $\neg C \mapsto \mathbf{T}$  (only possibility), hence (contrapositive)  $\neg B \mapsto \mathbf{T}$ , hence (contrapositive)  $\neg A \mapsto \mathbf{T}$ , i.e.  $A \mapsto \mathbf{F}$ ; thus, necessarily, for all the premises to be  $\mathbf{T}$ :  $E \mapsto \mathbf{T}$ . We conclude that the reasoning is correct.  $\square$

EXERCISE 3.10.–

a)

$$\begin{aligned} P \wedge (Q \Rightarrow R) \Rightarrow S &\longrightarrow \neg(P \wedge (Q \Rightarrow R)) \vee S \longrightarrow (\neg P \vee \neg(Q \Rightarrow R)) \vee S \\ &\longrightarrow (\neg P \vee \neg(\neg Q \vee R)) \vee S \longrightarrow (\neg P \vee (Q \wedge \neg R)) \vee S \\ &\longrightarrow ((\neg P \vee Q) \wedge (\neg P \vee \neg R)) \vee S \longrightarrow (\neg P \vee Q \vee S) \wedge (\frac{1}{2}\neg P \vee \neg R \vee S) \end{aligned}$$

b)

$$(P \vee \neg Q) \Rightarrow R \longrightarrow \neg(P \vee \neg Q) \vee R \longrightarrow (\neg P \wedge Q) \vee R]$$

c)

No, for example:

$(P \vee \neg Q \vee R)$ : **cnf** (a conjunct with three literals)

$(P) \vee (\neg Q) \vee (R)$ : **dnf** (three disjuncts, each with one literal)

d) No, it suffices to apply the distributivity of  $\wedge$  w.r.t.  $\vee$

% Note the analogy with Cartesian product:

$$\begin{aligned} (A \vee \neg B \vee C) \wedge (\neg D \vee E) \wedge (F \vee \neg G \vee H) &\longrightarrow (A \wedge \neg D \wedge F) \vee (A \wedge \neg D \wedge \neg G) \\ \vee (A \wedge \neg D \wedge H) \vee (A \wedge E \wedge F) \vee (A \wedge E \wedge \neg G) \vee (A \wedge E \wedge H) \vee (\neg B \wedge \neg D \wedge F) \\ \vee (\neg B \wedge \neg D \wedge \neg G) \vee \dots \vee (C \wedge E \wedge F) \vee (C \wedge E \wedge \neg G) \vee (C \wedge E \wedge H) \end{aligned}$$

(i.e.  $3 \times 2 \times 3 = 18$  disjuncts)

e)

No.

Consider the wff:

$$F : (P \Leftrightarrow Q) \wedge (Q \Leftrightarrow R) \wedge (R \Leftrightarrow P)$$

The wffs  $G$  and  $H$  below:

$$G : (\neg P \vee Q) \wedge (\neg Q \vee R) \wedge (\neg R \vee P)$$

$$H : (\neg Q \vee P) \wedge (\neg P \vee R) \wedge (\neg R \vee Q)$$

Are two cnfs of  $F$  and they are different. □

EXERCISE 3.11.–

$N$ : There is a unique norm to judge greatness in art.

$M$ :  $M$  is a great artist.

$G$ :  $G$  is a great artist.

$P$ :  $P$  is considered as a great artist.

$D$ :  $D$  is considered as a great artist.

$W$ :  $W$  is a great artist.

$K$ :  $K$  is a great artist.

$S$ :  $S$  is a great artist.

The following formalization seems “natural”:

$$1) N \Rightarrow \neg(M \wedge G)$$

$$2) P \vee D \Rightarrow \neg W$$

$$3) \neg W \Rightarrow \neg(K \vee S)$$

$$4) \neg G$$

$$5) D \wedge K$$

---


$$6) \neg N$$

*Forward chaining*: we try to evaluate the set of premises to **T** (in every possible way) and verify that all models of this set are also models of the conclusion.

( $\longrightarrow$  shows the evaluation sequence)

in (4)  $G \mapsto \mathbf{F}$  (only possibility)  $\longrightarrow$  in (5)  $D \mapsto \mathbf{T}$  (only possibility); in (5)  $K \mapsto \mathbf{T}$  (only possibility), hence in (3)  $W \mapsto \mathbf{T}$ ; hence, in (2)  $P \vee D$  must be evaluated to **F**: impossible.

As there is no other choice to evaluate the premises to **T**, we conclude that the premises are contradictory, and hence the reasoning is *trivially correct*.

*Backward chaining*: if we can evaluate the conclusion to **F** and the set of premises to **T**, then we refute the reasoning.

( $\longrightarrow$  indicates the evaluation sequence)

in (6)  $N \mapsto \mathbf{T} \longrightarrow$  in (1)  $G \mapsto \mathbf{F}$  or  $M \mapsto \mathbf{F} \longrightarrow$  in (5)  $D \mapsto \mathbf{T}$  and  $K \mapsto \mathbf{T}$  (only possibility)  $\longrightarrow W \mapsto \mathbf{T}$  (only possibility)  $\longrightarrow P \vee D$  must be evaluated to  $\mathbf{F}$ : impossible. Hence (of course!) the same conclusion, meaning that we cannot refute the reasoning which is correct.  $\square$

## EXERCISE 3.12.–

a) As we can evaluate the conclusion independently from the premises, we will be able to evaluate the conclusion to  $\mathbf{F}$  and the premises to  $\mathbf{T}$ . In general, the reasoning will therefore be incorrect.

*Particular case:* trivially correct reasonings (see exercise 3.9).

b) See proof of the following assertion exercise 3.30:

$S$ : set of clauses,  $L \in C \in S$ ,  $L$  pure

(\*)  $S$  is contradictory iff  $S \setminus \{C\}$  is contradictory

We shall prove that a reasoning is correct by proving that the set of premises together with the negation of the conclusion is contradictory. We can use property (\*) for this verification.

Example:

$$S = \{C_1, C_2, C_3, C_4\}$$

$$C_1 : P \vee Q \vee R$$

$$C_2 : \neg P \vee \neg Q$$

$$C_3 : \neg R \vee U$$

$$C_4 : \neg U \vee T$$

Step 1: we erase  $C_4$  ( $T$  pure)

Step 2: we erase  $C_3$  ( $U$  pure)

Step 3: we erase  $C_1$  ( $R$  pure)

Step 4: we erase  $C_2$  ( $\neg P$  and  $\neg Q$  pure)

$S$  is thus equivalent to  $\emptyset$ ; hence, non-contradictory (satisfiable). The reasoning from which  $S$  was generated was therefore incorrect.  $\square$

EXERCISE 3.13.– We use the following propositional symbols:

$C$ : Someone asked the house servant the question “...”.

$R$ : The house servant answered.

$E$ : Someone heard the house servant (the house servant was heard).

$Q$ : Someone saw the house servant.

$T$ : The house servant was busy polishing cutlery.

$U$ : The house servant was here on the day of the crime.

The reasoning can be formalized as follows:

$$1) C \Rightarrow R$$

$$2) R \Rightarrow E$$

$$3) \neg E$$

$$4) \neg Q \wedge \neg E \Rightarrow T$$

$$5) T \Rightarrow U$$

---


$$6) U$$

The reasoning is incorrect: the interpretation  $\{Q\}$ , i.e.  $Q$  evaluated to **T** and all other propositional symbols evaluated to **F** (see example 3.4) is a counter example.

Instead of verifying the correctness of the reasoning by using the same method as previously, we will do so in a way that is closer to the *syntactic* point of view of inference, similar to the inference rules used in mathematics, which will be treated in detail in section 3.3.

A major difference with the standard practice of mathematics is that we declare all the inference rules (elementary reasonings) that will be the only rules we will be allowed to use.

*Inference rules:*

$$\text{MP: } \frac{P \quad P \Rightarrow Q}{Q}$$

$$\text{and (intro): } \frac{P \quad Q}{P \wedge Q}$$

$$\text{and (elim): } \frac{P \wedge Q}{Q}$$

$$\text{and (reim): } \frac{P \wedge Q}{P}$$

or (intro):  $\frac{P}{P \vee Q}$

contrap:  $\frac{P \Rightarrow Q}{\neg Q \Rightarrow \neg P}$

disj. syl.:  $\frac{\neg P \quad P \vee Q}{Q}$

abd:  $\frac{Q \quad P \Rightarrow Q}{P}$  % Warning: **abd** is not a correct rule (see definition 3.12). It is used to discover premises in abduction (see section 8.3).

a) A trivial possibility: add  $U$  to the premises.

Other possibilities (closer to the intuitive notion of an explanation)

Forward chaining:

a1) We add  $\neg Q$  and we deduce:

7)  $T$  (3), (a1), (4), and MP

8)  $U$  (5), (7), MP

Other possibility:

7')  $\neg R$  (3), (2), contrap

8')  $\neg C$  (7'), (1), contrap

a2) We add  $\neg C \Rightarrow \neg Q$  (or  $Q \Rightarrow C$ ) and (a1), we obtain  $U$ .

Backward chaining:

7'')  $T$  (6), (5), **abd**

8'')  $\neg Q \wedge \neg E$  (7), (4), **abd**

a3) We add  $\neg Q$  and we obtain  $U$  as above.

b) There are infinitely many solutions (redundant intermediate chains that can be removed).

c) No: either the added premises are in contradiction with the existing premises, in which case the reasoning is trivially correct, or we carry out the same proof, as we did before the addition of the premises.

The key remark here is that a model of a set of formulas is a model of all the formulas in the set.

Assume  $P_1, P_2, \dots, P_n \models C$  and:

(\*)  $P_1, P_2, \dots, P_n, Q \not\models C$  %  $Q$  can be any formula, in particular, the “strange” case where  $Q : \neg C$  (because it is impossible to evaluate the premises to **T** and  $C$  to **F**).

(\*) means that there exists an interpretation  $\mathcal{I}$  that is a model of  $\{P_1, P_2, \dots, P_n, Q\}$  and a counter-model of  $C$ .

But a model of  $\{P_1, P_2, \dots, P_n, Q\}$  is also (by definition) a model of  $\{P_1, P_2, \dots, P_n\}$ , and every model of  $\{P_1, P_2, \dots, P_n\}$  is a model of  $C$ ; hence, (\*) is impossible.

REMARK 12.4.– In the formal system (see definition 3.9) whose inference rules are those above, it is possible to prove the *ex falso quodlibet* principle: “from a contradiction we can deduce any conclusion” (see also exercise 3.26).

- |                      |                     |   |
|----------------------|---------------------|---|
| 1) $P \wedge \neg P$ | premise             |   |
| 2) $P$               | (1), and (relim)    |   |
| 3) $P \vee Q$        | (2), or (intro)     |   |
| 4) $\neg P$          | (1), and (lelim)    |   |
| 5) $Q$               | (3), (4) disj. syl. | □ |

EXERCISE 3.14.– *only if*)

Trivial. By definition:

$A \models B$  means: *every model of  $A$  is a model of  $B$* . This means that  $A \Rightarrow B$  cannot be evaluated to **F**, i.e.:

$$\models A \Rightarrow B$$

*if*

$\models A \Rightarrow B$  means that *it is impossible to have* interpretations that evaluate  $A$  to **T** and  $B$  to **F**, in other words:

$$A \models B \quad \square$$

EXERCISE 3.15.–

a) Particular case of exercise 3.14.

b)  $H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$  iff  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  is unsatisfiable.

only if)

All models of  $H_1 \wedge H_2 \wedge \dots \wedge H_n$  are models of  $C$  (definition of  $\models$ ), hence, are counter-models of  $\neg C$ . If  $H_1 \wedge H_2 \wedge \dots \wedge H_n$  is evaluated to  $\mathbf{F}$ , then so is  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$ . If  $H_1 \wedge H_2 \wedge \dots \wedge H_n$  is evaluated to  $\mathbf{T}$ , then  $\neg C$  is evaluated to  $\mathbf{F}$ , and so is  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$ . Conclusion:  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  is unsatisfiable.

if)

It suffices to restrict ourselves to the interpretations of  $H_1 \wedge H_2 \wedge \dots \wedge H_n \wedge \neg C$  that are models of  $H_1 \wedge H_2 \wedge \dots \wedge H_n$ .

If  $\mathcal{M}$  is a model of  $H_1 \wedge H_2 \wedge \dots \wedge H_n$ , then by definition of unsatisfiability,  $\mathcal{E}(\neg C, \mathcal{M}) = \mathbf{F}$ , hence  $\mathcal{E}(C, \mathcal{M}) = \mathbf{T}$ ; therefore,  $H_1 \wedge H_2 \wedge \dots \wedge H_n \models C$ .  $\square$

EXERCISE 3.16.–

Answer: (b)

We prove so by *reductio ad absurdum*.

Assume  $A \models B$ , hence:

$A \models A \wedge B$  (definition of  $\models$ )

but  $A \wedge B \models C$  (hypothesis)

thus (transitivity of  $\models$ )

$A \models C$  contradiction, therefore:

$A \not\models B$

$\square$

EXERCISE 3.17.–

True.

The reasonings will be of the following form.

Premises:  $S_1 = \{f_{k_1}, f_{k_2}, \dots, f_{k_{n-1}}\}$

Conclusion:  $\neg f_l$

where  $k_j \neq l$  for  $1 \leq j \leq n-1$  and  $1 \leq l \leq n$ .

Indeed, a  $S$  is minimally unsatisfiable,  $S_1 \subsetneq S$  is satisfiable and every model of  $S_1$  must be a counter-model of  $f_l$  (a  $S$  is unsatisfiable), i.e.:

$$S_1 \models \neg f_l. \quad \square$$

REMARK 12.5.— This property can be viewed as a generalization of the proof technique that consists in proving the contrapositive.

$$\text{If we have proved } P_1 \wedge P_2 \dots \wedge P_n \models C$$

by proving:

$$\{P_1, P_2, \dots, P_n, \neg C\} \text{ unsat,}$$

then we have also proved that

$$\neg C \models \neg(P_1 \wedge P_2 \dots \wedge P_n)$$

(if  $\neg C$  is **T**, then  $(P_1 \wedge P_2 \dots \wedge P_n)$  must be **F**, otherwise the set would be satisfiable):

i.e.:

$$\neg C \models \neg P_1 \vee \neg P_2 \dots \vee \neg P_n$$

or, in other words:

$$\neg C \wedge P_1 \wedge P_2 \dots \wedge P_{n-1} \models \neg P_n. \quad \square$$

EXERCISE 3.18.—

a) The ideas and remarks that will enable us to construct the interpolant are the following:

1) Each line<sup>1</sup> of the truth table of a formula  $\mathcal{F}$  is an interpretation of  $\mathcal{F}$  (the truth table enumerates all the interpretations of  $\mathcal{F}$ ). We shall mention *interpretation* with the same meaning as *line in the truth table*.

2) The interpretations of interpolant  $\mathcal{C}$  to be constructed will, in general, be partial interpretations of  $\{\mathcal{A}, \mathcal{B}\}$  (except in the case in which  $\text{Propset}(\mathcal{A}) = \text{Propset}(\mathcal{B})$ ).

3) Key remark. Given an interpretation  $\mathcal{J}$  of  $\mathcal{C}$ , does there exist an interpretation  $\mathcal{I}$  of  $\{\mathcal{A}, \mathcal{B}\}$ , that is an extension of  $\mathcal{J}$  and such that:

---

<sup>1</sup> We exclude the truth value assigned to the corresponding formula from the line.



$$\mathcal{A} \not\models_I \mathcal{B}$$

As, by hypothesis,  $\mathcal{A} \models \mathcal{B}$ , i.e. every interpretation satisfying  $\mathcal{A}$  also satisfies  $\mathcal{B}$ , such an interpretation  $I$  does not exist.

4) We want  $\mathcal{A} \models \mathcal{C}$ ; hence, if line 1 of the truth table of  $\mathcal{A}$  evaluates  $\mathcal{A}$  to **T** and  $l$  contains line  $l_C$  of the truth table of  $\mathcal{C}$ , then we will evaluate line  $l_C$  of  $\mathcal{C}$  to **T**.

5) We want  $\mathcal{C} \models \mathcal{B}$ ; hence, if line  $l$  of the truth table of  $\mathcal{B}$  evaluates  $\mathcal{B}$  to **F** and contains line  $l_C$  of the truth table of  $\mathcal{C}$ , then we will evaluate line  $l_C$  of  $\mathcal{C}$  to **F**.

6) There cannot be any conflict between steps (4) and (5) (see remark 3).

7) For the cases in which line  $l$  evaluates  $\mathcal{A}$  to **F** (respectively,  $\mathcal{B}$  to **T**), we can arbitrarily choose to assign values **T** or **F** to line  $l_C$  of  $\mathcal{C}$ , which means that, in general, there are several possible interpolants ( $2^n$ , where  $n$  is the number of lines from which we can choose).

8) Points (1)–(7) above are about the truth table of  $\mathcal{C}$ . The method described in exercise 3.2 enables us to construct the dnf (or cnf) of  $\mathcal{C}$ .

(1)–(8) above give the algorithm to construct  $\mathcal{C}$  and prove its correctness at the same time.

$$\text{b) } \text{Propset}(\mathcal{A}) = \text{Propset}(\mathcal{B}) = \text{Propset}(\mathcal{C}) = \{A, B, C\}$$

$A$	$B$	$C$	$\overbrace{A \Leftrightarrow (B \vee C)}^{\mathcal{A}}$	$\overbrace{(A \wedge \neg B) \Rightarrow C}^{\mathcal{B}}$	$\mathcal{C}$
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b> or <b>F</b> (arbitrarily)
<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b> or <b>F</b> (arbitrarily)
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b> or <b>F</b> (arbitrarily)
<b>F</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>

By applying the method of exercise 3.2, we obtain the  $8(=2^3)$  possible interpolants:

$$C_1 : (A \wedge B \wedge C) \vee (A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$$

$$C_2 : C_1 \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C)$$

$$C_3 : C_1 \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge B \wedge \neg C)$$

$$C_4 : C_1 \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge C)$$

$$C_5 : C_1 \vee (\neg A \wedge B \wedge C)$$

$$C_6 : C_1 \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C)$$

$$C_7 : C_1 \vee (\neg A \wedge B \wedge \neg C)$$

$$C_8 : C_1 \vee (\neg A \wedge \neg B \wedge C)$$

c)  $Propset(\mathcal{A}) = \{A, B, C\}$ ;  $Propset(\mathcal{B}) = \{B, C, D\}$ ;  $Propset(\mathcal{C}) = \{B, C\}$

$A \ B \ C \ D \ \neg D$	$\overbrace{(\neg A \wedge \neg B) \wedge (A \Leftrightarrow C)}^{\mathcal{A}}$	$\overbrace{(C \Rightarrow B) \wedge (\neg D \vee \neg C)}^{\mathcal{B}}$
<b>T T T T F</b>	<b>F</b>	<b><u>F</u></b>
<b>T T T F T</b>	<b>F</b>	<b>T</b>
<b>T T F T F</b>	<b>F</b>	<b>T</b>
<b>T T F F T</b>	<b>F</b>	<b>T</b>
<b>T F T T F</b>	<b>F</b>	<b><u>F</u></b>
<b>T F T F T</b>	<b>F</b>	<b><u>F</u></b>
<b>T F F T F</b>	<b>F</b>	<b>T</b>
<b>T F F F T</b>	<b>F</b>	<b>T</b>
<b>F T T T F</b>	<b>F</b>	<b><u>F</u></b>
<b>F T T F T</b>	<b>F</b>	<b>T</b>
<b>F T F T F</b>	<b>F</b>	<b>T</b>
<b>F T F F T</b>	<b>F</b>	<b>T</b>
<b>F F T T F</b>	<b>F</b>	<b><u>F</u></b>
<b>F F T F T</b>	<b>F</b>	<b><u>F</u></b>
<b>F F F T F</b>	<b><u>T</u></b>	<b>T</b>
<b>F F F F T</b>	<b><u>T</u></b>	<b>T</b>

$B \ C$	$\mathcal{C}_1$	$\mathcal{C}_2$
<b>T T</b>	<b>F</b>	<b>F</b>
<b>T F</b>	<b><u>T</u></b>	<b>F</b>
<b>F T</b>	<b>F</b>	<b>F</b>
<b>F F</b>	<b>T</b>	<b>T</b>

$$\mathcal{C}_1 : (B \wedge \neg C) \vee (\neg B \wedge \neg C) \text{ equiv } \neg C$$

$$\mathcal{C}_2 : \neg B \wedge \neg C$$

□

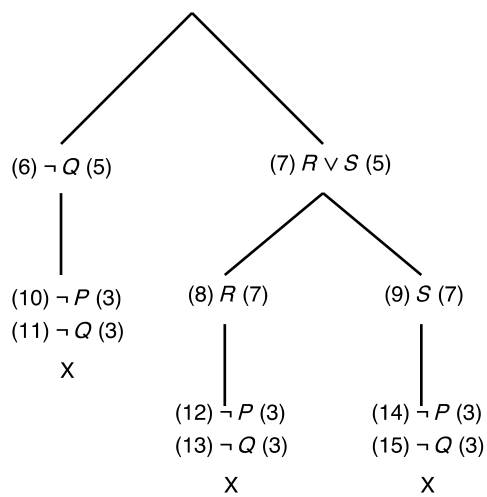
EXERCISE 3.19.–

Tree 1

- 1)  $\neg((P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q))$
- 2)  $P \wedge (Q \Rightarrow (R \vee S))$
- 3)  $\neg(P \vee Q)$
- 4)  $P$
- 5)  $Q \Rightarrow (R \vee S)$
- |
- 6)  $\neg P$  (3)
- 7)  $\neg Q$  (3)
- 8)  $\times$  (6), (4)

Tree 2

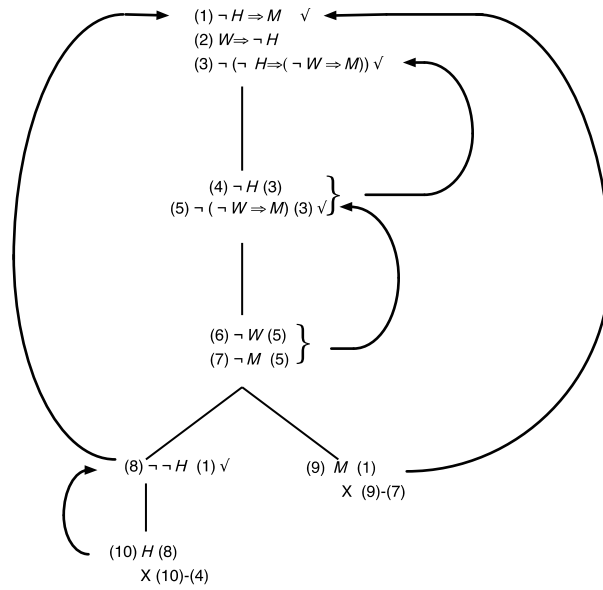
- 1)  $\neg((P \wedge (Q \Rightarrow (R \vee S))) \Rightarrow (P \vee Q))$
- 2)  $P \wedge (Q \Rightarrow (R \vee S))$
- 3)  $\neg(P \vee Q)$
- 4)  $P$
- 5)  $Q \Rightarrow (R \vee S)$



□

EXERCISE 3.20.–

a)



Correct reasoning

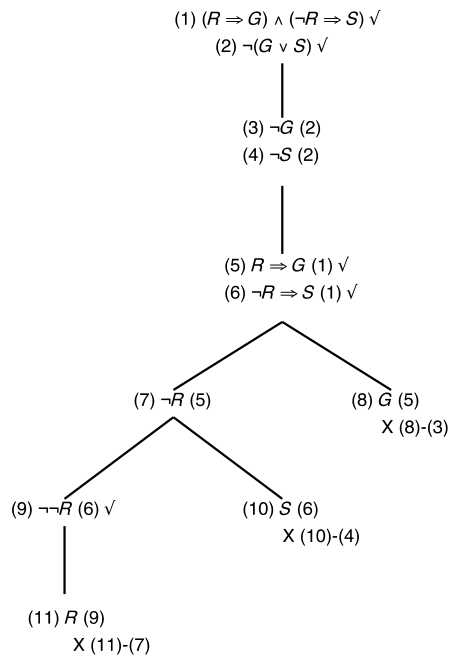
The }'s, the ✓'s, and the arrows correspond to the operations:

$$\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\}) \cup \{f_i^j\} \text{ and}$$

$$\mathcal{F} \leftarrow (\mathcal{F} \setminus \{f_i\})$$

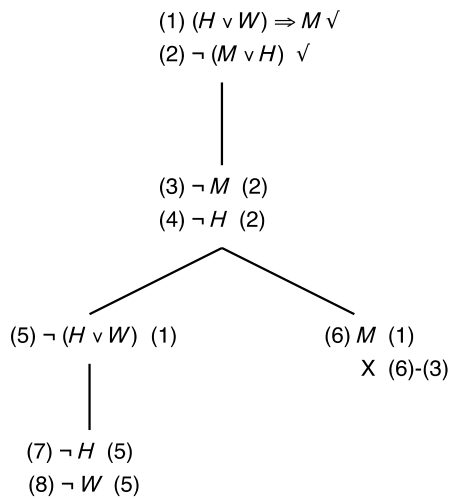
of the algorithm SEMANTIC TABLEAUX (PL).

b)



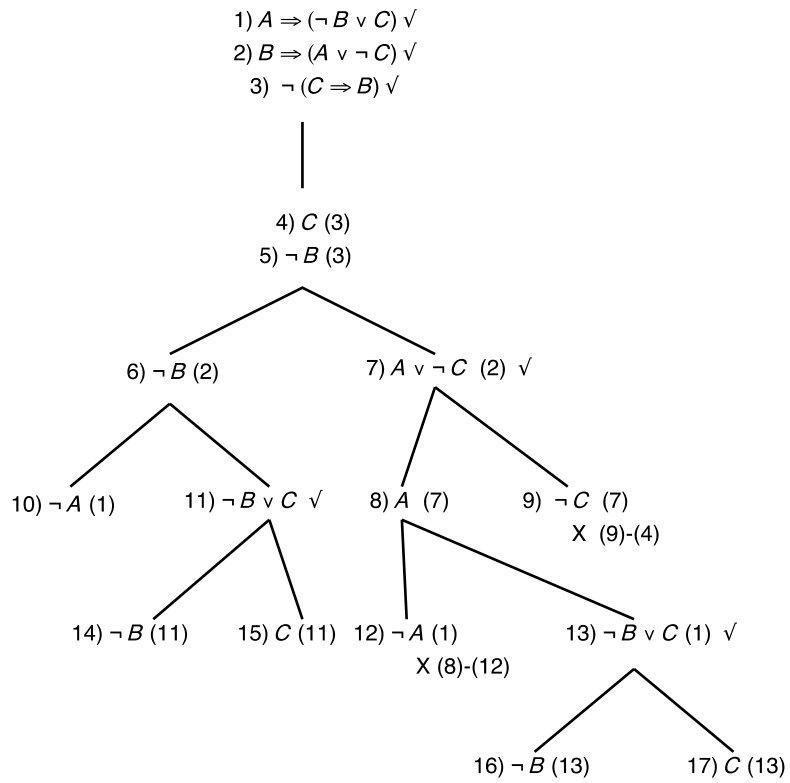
Correct reasoning

c)



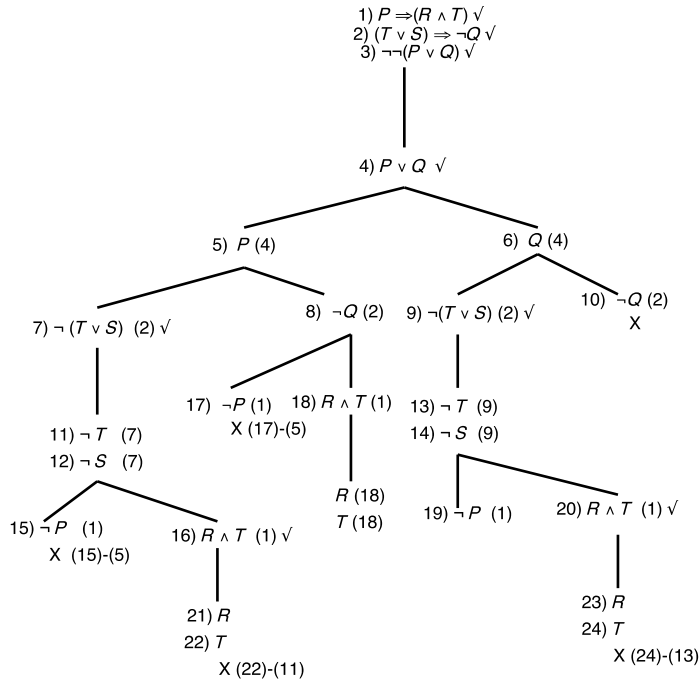
Incorrect reasoning, counter example:  $\{\neg H, \neg M, \neg W\}$

d)



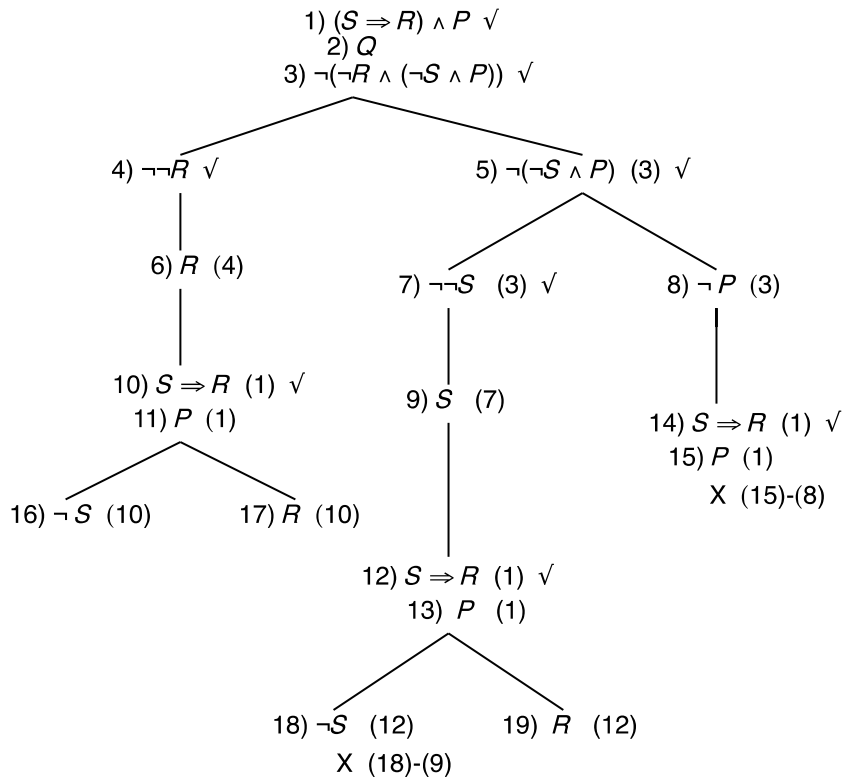
Incorrect reasoning, counter examples:  $\{\neg A, \neg B, C\}$ ,  $\{\neg B, C\}$ , and  $\{A, \neg B, C\}$

e)



Incorrect reasoning, counter examples:  $\{P, \neg Q, R, T\}$  and  $\{\neg P, Q, \neg S, \neg T\}$

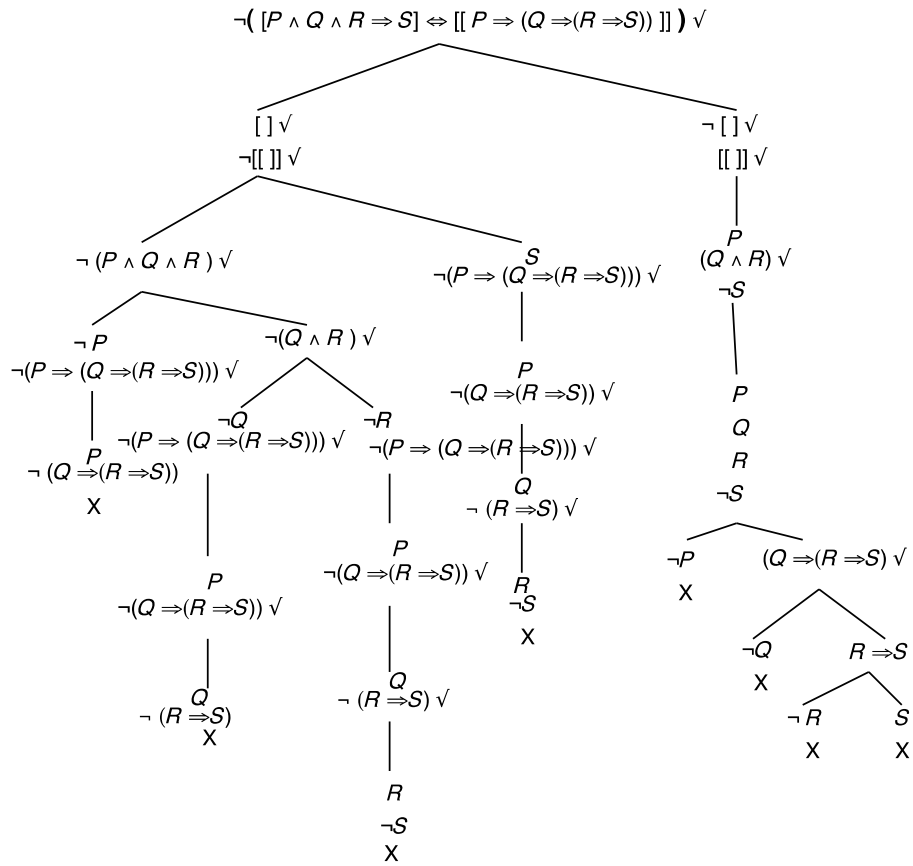
f)



Incorrect reasoning, counter examples:  $\{P, Q, R, \neg S\}$ ,  $\{P, Q, R\}$ , and  $\{P, Q, R, S\}$



g)

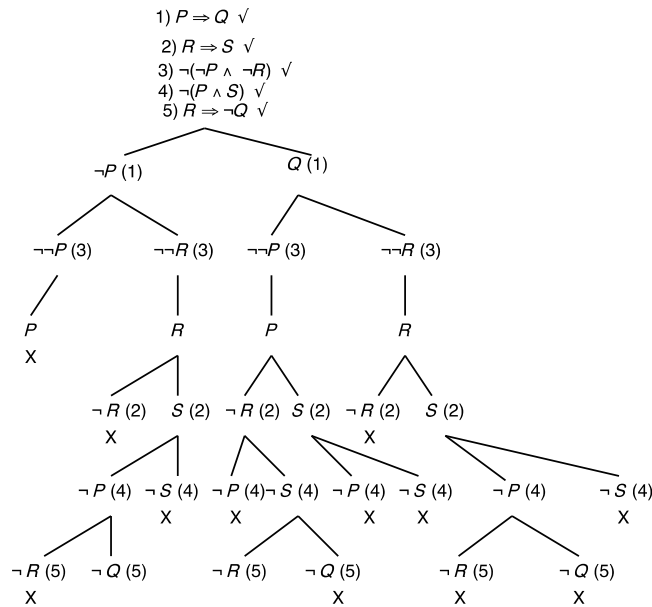


Thus, the initial formula is valid.

□

EXERCISE 3.21.–

a)



The set of formulas is **satisfiable**, with models  $\{\neg P, R, S, \neg Q\}$  and  $\{Q, P, \neg R, \neg S\}$

b)

Although this is not necessary at all, sometimes, a preprocessing can enable us to simplify the problem under consideration. We illustrate this by applying the purity principle, i.e. the removal, in a set of clauses, of those clauses containing pure literals, since such an operation preserves the (un)satisfiability of the set (see exercise 3.30).

We must thus transform  $S_2$  into an equivalent set of clauses.

- 1)  $\neg P$
- 2)  $\neg R \Rightarrow W \quad \longrightarrow \quad \neg R \vee W \quad \text{Step 2: clause deleted, } W \text{ pure}$
- 3')  $Q \vee T \vee \neg P$
- 3)  $Q \vee (\neg T \Rightarrow \neg(P \vee U)) \longrightarrow \langle$ 
  - 3'')  $Q \vee T \vee \neg U$
  - 4')  $P \vee U$

4)  $\neg P \Rightarrow (U \wedge \neg R) \quad \longrightarrow \langle$

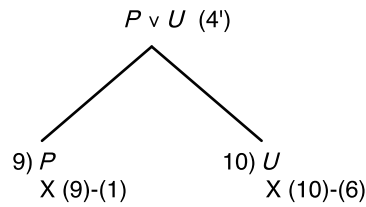
4'')  $P \vee \neg \neg R$  Step 3: clause deleted,  $\neg R$  pure

5)  $\neg Q$

6)  $\neg U$

7)  $\neg T$

8)  $\neg R \Rightarrow S \quad \longrightarrow \quad \neg R \vee S$  Step 1: clause deleted,  $S$  pure



The set of formulas is unsatisfiable. □

EXERCISE 3.22.– We give two reasons.

1) If they were defined as functions:

– they would either be partial functions, for example:

$$\text{MP: } \frac{A \quad B \Rightarrow C}{\perp}$$

– or we use a trick:

$$\frac{A \quad B \Rightarrow C}{A}$$

2) We would like to have, for example:

$$\frac{P \wedge Q}{P}, \text{ but also:}$$

$$\frac{P \wedge Q}{Q}$$

This inference rule is clearly not a function, as it gives two distinct values for the same argument.

In general, it is impossible to consider inference rules as functions without any problem arising. □

EXERCISE 3.23.– We want to prove:

If  $\Gamma \vdash_{S_1} A \Rightarrow B$ , then  $\Gamma, A \vdash_{S_1} B$

PROOF.– By definition, there exists a deduction of  $A \Rightarrow B$  starting from  $\Gamma$ :

$\Gamma$

$\vdots$

$A \Rightarrow B$

if we add  $A$  to the hypotheses, we obtain  $B$  by MP on  $A$  and  $A \Rightarrow B$ , i.e.

$\Gamma, A \vdash_{S_1} B$ . □

EXERCISE 3.24.–

a) **MP** is correct: every model of  $A$  and  $A \Rightarrow B$  is necessarily a model of  $B$  (otherwise  $B$  would be **F**, by definition of  $\Rightarrow$ ).

We can easily verify that all the axiom schemas are tautologies (valid wffs). Hence, by definition of a valid wff and by induction on the number of steps in the proof, we conclude that every theorem of  $S_1$  is a valid wff.

b) Assume that there exists  $A \in \mathcal{L}$  such that  $\vdash_{S_1} A$  and  $\vdash_{S_1} \neg A$ .

As shown in (a) above, every theorem of  $S_1$  is a valid wff. The negation of a valid wff is not a valid wff, hence it is not a theorem (contrapositive).

Therefore, such a wff  $A$  cannot exist.

c) Truth tables (semantic tableaux, the Davis and Putnam method, etc.) permit us to decide whether a wff is valid or not. As we admitted that  $S_1$  is adequate, we have a decision procedure for  $S_1$ .

REMARK 12.6.– The soundness proved in (a) is not sufficient to prove (c). Indeed, the former says: *if not valid then not a theorem*. But if adequacy (completeness) is not guaranteed and the wff under consideration is valid, then we cannot be sure that it is a theorem of  $S_1$  (it could be a valid wff that is not captured as a theorem by the formal system). □

## EXERCISE 3.25.–

a)

- 1)  $(\neg A \Rightarrow \neg A) \Rightarrow ((\neg A \Rightarrow A) \Rightarrow A)$  (A3)  $B \leftarrow A$   
 2)  $\neg A \Rightarrow \neg A$  Example 3.9, page 79,  $A \leftarrow \neg A$

We can always use a theorem that was already proved. The justification is simple: we copy its proof at the beginning of the proof that uses it. The proof thus obtained respects the definition of a proof.

- 3)  $(\neg A \Rightarrow A) \Rightarrow A$  (1), (2), and MP

b) We give two deductions, one of which uses the deduction (meta-) theorem (abbreviated as DT).

i) *Without using the DT*

- 1)  $A \Rightarrow (B \Rightarrow C)$  hyp.  
 2)  $B$  hyp.  
 3)  $(A \Rightarrow B) \Rightarrow (A \Rightarrow C)$  (1), (A2), MP  
 4)  $B \Rightarrow (A \Rightarrow B)$  (A1)  $A \leftarrow B, B \leftarrow A$   
 5)  $A \Rightarrow B$  (2), (4), MP  
 6)  $A \Rightarrow C$  (3), (5), MP

ii) *Using the DT*

- 1)  $A \Rightarrow (B \Rightarrow C)$  hyp.  
 2)  $B$  hyp.  
 3)  $A$  add hyp.  
 4)  $B \Rightarrow C$  (1), (3), MP  
 5)  $C$  (2), (4), MP  
 6)  $A \Rightarrow C$  (3), (5), DT

c)

- 1)  $A \Rightarrow B$  hyp.  
 2)  $B \Rightarrow C$  hyp.  
 3)  $A$  hyp.  
 4)  $B$  (1), (3), MP  
 5)  $C$  (2), (4), MP



f)

- |  |  |
|--|--|
| 1) $(\neg A \Rightarrow \neg\neg A) \Rightarrow ((\neg A \Rightarrow \neg A) \Rightarrow A)$ | (A3), $B \leftarrow A$ , $A \leftarrow \neg A$                     |
| 2) $\neg A \Rightarrow \neg A$   | Example 3.9, page 79, $A \leftarrow \neg A$                        |
| 3) $(\neg A \Rightarrow \neg\neg A) \Rightarrow A$   | (1), (2), (b) above, $B \leftarrow \neg A$ , $C \leftarrow \neg A$ |
| 4) $\neg\neg A \Rightarrow (\neg A \Rightarrow \neg\neg A)$                                  | (A1), $A \leftarrow \neg\neg A$ , $B \leftarrow \neg A$            |
| 5) $\neg\neg A$  | add. hyp.  |
| 6) $\neg A \Rightarrow \neg\neg A$   | (4), (5), MP   |
| 7) $A$   | (3), (6), MP   |
| 8) $\neg\neg A \Rightarrow A$  | (5), (7), DT   |

*Other proof:*

- |  |   |
|--|---|
| 1) $\neg\neg A$  | add.hyp.  |
| 2) $\neg\neg A \Rightarrow (\neg A \Rightarrow \neg\neg A)$                                  | (A1), $A \leftarrow \neg\neg A$ , $B \leftarrow \neg A$ |
| 3) $\neg A \Rightarrow \neg\neg A$   | (1), (2), MP  |
| 4) $(\neg A \Rightarrow \neg\neg A) \Rightarrow ((\neg A \Rightarrow \neg A) \Rightarrow A)$ | (A3), $(B) \leftarrow (A)$ , $(A) \leftarrow \neg(A)$   |
| 5) $(\neg A \Rightarrow \neg A) \Rightarrow A$   | (3), (4), MP  |
| 6) $\neg A \Rightarrow \neg A$   | Example 3.9, $A \leftarrow \neg A$                      |
| 7) $A$   | (5), (6), MP  |
| 8) $\neg\neg A \Rightarrow A$  | (1), (7), DT  |

*Another one:*

- |  |   |
|--|---|
| 1) $(\neg A \Rightarrow \neg\neg A) \Rightarrow ((\neg A \Rightarrow \neg A) \Rightarrow A)$ | (A3), $B \leftarrow A$ , $A \leftarrow \neg A$          |
| 2) $\neg A \Rightarrow \neg A$   | Example 3.9, $A \leftarrow \neg A$                      |
| 3) $(\neg A \Rightarrow \neg\neg A) \Rightarrow A$   | (1), (2), (b) above                                     |
| 4) $\neg\neg A \Rightarrow (\neg A \Rightarrow \neg\neg A)$                                  | (A1), $A \leftarrow \neg\neg A$ , $B \leftarrow \neg A$ |
| 5) $(\neg\neg A \Rightarrow A)$  | (4), (3), (e)   |

g)

- |  |                                     |
|--|-------------------------------------|
| 1) $(\neg\neg\neg A \Rightarrow \neg A) \Rightarrow ((\neg\neg\neg A \Rightarrow A) \Rightarrow \neg\neg A)$ | (A3), $B \leftarrow \neg\neg A$     |
| 2) $\neg\neg\neg A \Rightarrow \neg A$   | f), $A \leftarrow \neg A$           |
| 3) $(\neg\neg\neg A \Rightarrow A) \Rightarrow \neg\neg A$   | (1), (2), MP                        |
| 4) $A \Rightarrow (\neg\neg\neg A \Rightarrow A)$  | (A1), $B \leftarrow \neg\neg\neg A$ |
| 5) $(A \Rightarrow \neg\neg A)$  | (4), (3), (e)                       |

h)

- |   |           |
|---|-----------|
| 1) $A \Rightarrow B, B \Rightarrow C, A \vdash C$   | (c) above |
| 2) $A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C$                                | (1), DT   |
| 3) $A \Rightarrow B \vdash (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$                 | (2), DT   |
| 4) $\vdash (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$ | (3), DT   |

*Other deduction:*

- |  |  |
|--|--|
| 1) $A \Rightarrow B$   | add.hyp.   |
| 2) $B \Rightarrow C$   | add.hyp.   |
| 3) $(B \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))$                                 | (A1), $A \leftarrow B \Rightarrow C, B \leftarrow A$ |
| 4) $A \Rightarrow (B \Rightarrow C)$   | (2), (3), MP   |
| 5) $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$ | (A2)   |
| 6) $(A \Rightarrow B) \Rightarrow (A \Rightarrow C)$   | (4), (5), MP   |
| 7) $A \Rightarrow C$   | (1), (6), MP   |
| 8) $A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C$   | (1), (2), (7)  |
| 9) $A \Rightarrow B \vdash (B \Rightarrow C) \Rightarrow (A \Rightarrow C)$                          | (8), DT  |
| 10) $\vdash (A \Rightarrow B) \Rightarrow ((B \Rightarrow C) \Rightarrow (A \Rightarrow C))$         | (9), DT  |

i)

- |   |           |
|---|-----------|
| 1) $A \Rightarrow (B \Rightarrow C), B \vdash A \Rightarrow C$                              | (b) above |
| 2) $A \Rightarrow (B \Rightarrow C) \vdash B \Rightarrow (A \Rightarrow C)$                 | (1), DT   |
| 3) $\vdash (A \Rightarrow (B \Rightarrow C)) \Rightarrow (B \Rightarrow (A \Rightarrow C))$ | (2), DT   |

□

## EXERCISE 3.26.–

We must prove:

 $S_1$  is consistent w.r.t. negation if  $S_1$  is absolutely consistent.*only if*We prove the contrapositive. We assume that  $\tau = \mathcal{L}$ , i.e. that every wff is a theorem. In particular: $\vdash_{S_1} A$  and  $\vdash_{S_1} \neg A$ hence,  $S_1$  is not consistent w.r.t. negation.



if

We prove the contrapositive. We thus assume that there exists  $A \in \mathcal{L}$  such that:

$\vdash_{S_1} A$  and  $\vdash_{S_1} \neg A$

We first prove:

$A, \neg A \vdash_{S_1} B$

1) $A$	theorem assumed to be proved
2) $\neg A$	theorem assumed to be proved
3) $A \Rightarrow (\neg B \Rightarrow A)$	(A1), $B \leftarrow \neg B$ , $A \leftarrow A$
4) $\neg A \Rightarrow (\neg B \Rightarrow \neg A)$	(A1), $B \leftarrow \neg B$ , $A \leftarrow \neg A$
5) $\neg B \Rightarrow A$	(1), (3), MP
6) $\neg B \Rightarrow \neg A$	(2), (4), MP
7) $(\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B)$	(A3)
8) $(\neg B \Rightarrow A) \Rightarrow B$	(6), (7), MP
9) $B$	(5), (8), MP

Hence, as  $B$  can be replaced by any wff, we conclude that every wff is a theorem.  $\square$

EXERCISE 3.27.-

a)

1) $Q \Rightarrow (\neg P \vee Q)$	(A2)
2) $Q \Rightarrow (P \Rightarrow Q)$	(1), (D1)

b)

1) $(\neg P \vee \neg P) \Rightarrow \neg P$	(A1)
2) $(P \Rightarrow \neg P) \Rightarrow \neg P$	(1), (D1)

c)

1) $(\neg P \vee \neg Q) \Rightarrow (\neg Q \vee \neg P)$	(A3)
2) $(P \Rightarrow \neg Q) \Rightarrow (Q \Rightarrow \neg P)$	(1), (D1)

d)

- 1)  $(Q \Rightarrow R) \Rightarrow ((\neg P \vee Q) \Rightarrow (\neg P \vee R))$  (A4)  
 2)  $(Q \Rightarrow R) \Rightarrow ((P \Rightarrow Q) \Rightarrow (P \Rightarrow R))$  (1), (D1)

e)

- 1)  $P \Rightarrow (P \vee P)$  (A2)

f)

- 1)  $((P \vee P) \Rightarrow P) \Rightarrow ((P \Rightarrow (P \vee P)) \Rightarrow (P \Rightarrow P))$  (d), above :  
 $Q \leftarrow P \vee P; R \leftarrow P$   
 2)  $(P \vee P) \Rightarrow P$  (A1)  
 3)  $(P \Rightarrow (P \vee P)) \Rightarrow (P \Rightarrow P)$  (1), (2), MP  
 4)  $P \Rightarrow (P \vee P)$  (e), above  
 5)  $P \Rightarrow P$  (3), (4), MP

g)

- 1)  $P \Rightarrow P$  (f), above  
 2)  $\neg P \vee P$  (1), (D1)  
 3)  $(\neg P \vee P) \Rightarrow (P \vee \neg P)$  (A3)  
 4)  $P \vee \neg P$  (2), (3), MP

h)

- 1)  $\neg P \vee \neg\neg P$  (g), above :  $P \leftarrow \neg P$   
 2)  $P \Rightarrow \neg\neg P$  (1), (D1)

□

EXERCISE 3.28.-

a) Yes, by verifying that:

(B2) is the same axiom schema as (A1) (in  $\mathcal{S}_1$ );

(B4) is the same axiom schema as (A2) (in  $\mathcal{S}_1$ );

and by taking remark 3.24 into account.

b)

$$1) (\neg A \Rightarrow (A \Rightarrow \neg\neg A)) \Rightarrow (((\neg\neg A \Rightarrow (A \Rightarrow \neg\neg A)) \Rightarrow (A \Rightarrow \neg\neg A)))$$

$$B3, A \leftarrow \neg A, B \leftarrow A \Rightarrow \neg\neg A$$

$$2) \neg A \Rightarrow (A \Rightarrow \neg\neg A)$$

$$B1, B \leftarrow \neg\neg A$$

$$3) (\neg\neg A \Rightarrow (A \Rightarrow \neg\neg A)) \Rightarrow (A \Rightarrow \neg\neg A)$$

(1), (2), MP

$$4) \neg\neg A \Rightarrow (A \Rightarrow \neg\neg A)$$

$$B2, B \leftarrow \neg\neg A$$

$$5) A \Rightarrow \neg\neg A$$

(3), (4), MP. □

EXERCISE 3.29.–

a) We will say that two formal systems with the same language (or with languages that can be formally translated from one to the other) are equivalent iff they have the same set of theorems.

b) Consider:

- the formal system  $S_i : \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$ ;
- a subset of axioms  $\mathcal{X}$  ( $\mathcal{X} \subseteq \mathcal{A}$ );
- the formal system  $S_j : \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \setminus \mathcal{X} \rangle$ .

$\mathcal{X}$  is independent iff  $\not\vdash_{S_j} x$ , for all  $x \in \mathcal{X}$ .

For example, the formal system  $S'_1 : \langle \mathcal{L}, \mathcal{R}, \mathcal{A}' \rangle$  (see (c) below), which differs from  $S_1$  only in the set of axioms, is not independent.

$$c) \mathcal{A}' = \{(A1), (A2), (A3), A \Rightarrow A\}$$

(as  $\vdash_{S_1} A \Rightarrow A$ ; see example 3.9)

d) (analogous to (b))

Consider:

- the formal system  $S_i : \langle \mathcal{L}, \mathcal{R}, \mathcal{A} \rangle$ ;
- a subset of inference rules  $\mathcal{Y}$  ( $\mathcal{Y} \subseteq \mathcal{R}$ );
- the formal system  $S_j : \langle \mathcal{L}, \mathcal{R} \setminus \mathcal{Y}, \mathcal{A} \rangle$ .

$\mathcal{Y}$  is independent iff there exists  $A \in \mathcal{L}$  and  $\vdash_{S_i} A$ , such that  $\not\vdash_{S_j} A$ .

e) Find (it is not guaranteed that this will succeed) a property  $P$  such that:

- 1) the elements of  $\mathcal{A} \setminus \mathcal{X}$  have property  $P$ ;
- 2) the rules of  $\mathcal{R}$  preserve property  $P$ ;
- 3) the elements of  $\mathcal{X}$  do not have property  $P$ .

REMARK 12.7.– This technique was used to prove the independence of the famous “parallel postulate”, which led to non-Euclidean geometry.

Interpretations that are models of the other axioms of Euclidean geometry and counter-models of the parallel postulate have been found.  $\square$

EXERCISE 3.30.–

**(R-0)**

Without loss of generality, we assume that  $S$  contains a unique tautological clause  $T$ .

Let  $S_1 = S \setminus \{T\}$

$S$  is unsatisfiable iff  $S_1$  is satisfiable.

*if*

We prove the contrapositive:

$S$  satisfiable.

By definition, every subset of a satisfiable set is satisfiable; hence,  $S_1$  is satisfiable.

*only if*

We prove the contrapositive:

$S_1$  is sat, with model, say,  $M$ .

As  $T$  is satisfied by every interpretation, it must be satisfied by  $M$ ; hence,  $S$  is sat.

**(R-1a)** Trivial. No interpretation can make both  $L$  and  $L^c$  **T**.

**(R-1b)** (To simplify the notation, we assume there is only one unit clause  $\{L\}$  and only one clause of the form  $\{L^c\} \cup \alpha$ . Of course, the reasoning can be repeated.

$$S_1 = ((S \setminus \{L\}) \setminus (\{L^c\} \cup \alpha)) \cup \{\alpha\}$$

$S$  unsat iff  $S_1$  unsat.

*only if*

We prove the contrapositive.

Assume  $S_1$  is sat, let  $M_1$  be a model of  $S_1$ . Without loss of generality, we can assume  $L^c \notin M_1$  (otherwise, it can be removed from  $M_1$  without any consequence). There exists  $K \in \alpha$  such that  $K \in M_1$  (all the clauses in  $S_1$  must be evaluated to **T**).

$M = \{L\} \cup M_1$  is a model of  $S$ . Therefore,  $S$  is sat.

*if*

We prove the contrapositive:

Assume  $S$  is sat, let  $M$  be a model of  $S$ . Necessarily,  $L \in M$  (and therefore  $L^c \notin M$ ). There exists  $K \in \alpha$  such that  $K \in M$ ; thus,  $S_1$  is sat.

**(R2)**

We assume without loss of generality that there is a unique pure literal:

$L \in C \in S$  and  $L$ : pure

$$S_1 = S \setminus \{C\}$$

$S$  is unsat iff  $S_1$  is unsat.

*only if*

We prove the contrapositive.

Assume  $S_1$  is sat, let  $M_1$  be a model of  $S_1$ ,  $L^c \notin M_1$  (as  $L$  is pure).

$M_1 \cup \{L\}$  is a model of  $S$ , which is therefore sat.

*if*

We prove the contrapositive.

$S$  sat and  $S_1 \subsetneq S$ .  $S_1$  cannot be unsat because every model of  $S$  is a model of *all* the clauses in  $S$ . Thus,  $S_1$  is sat.

This rule is known as the *purity principle*.

**(R3)**

$$S_1 = \{C \setminus \{L\} \mid C \in S \text{ and } L^c \notin C\}$$

$$S_2 = \{C \setminus \{L^c\} \mid C \in S \text{ and } L \notin C\}$$

$$S \text{ unsat iff } (S_1 \text{ unsat and } S_2 \text{ unsat})$$

*only if*

We prove the contrapositive.

$S_1$  sat or  $S_2$  sat.

Without loss of generality, we assume that  $S_1$  is sat (we do not consider the case in which  $S_2$  is sat, as the proof is similar).

Let  $M_1$  be a model of  $S_1$ .  $L \notin M_1$  and there exists  $K \in M_1$ ,  $K \in C \setminus \{L\}$ .

$M = M_1 \cup \{L^c\}$  is a model of  $S$ . Hence  $S$  is sat.

*if*

We assume without loss of generality that there is only one clause containing  $L$  (say, the clause  $L \vee \alpha$ ) and only one clause containing  $L^c$  (say  $L^c \vee \beta$ ).

We prove the contrapositive.

$S$  is satisfiable, let  $M$  be a model of  $S$ . Assume  $L \in M$ ; hence,  $L^c \notin M$ . As  $M$  is a model of *all* the clauses in  $S$ , there exists a  $K \in \beta$  such that  $K \in M$ ; hence,  $S_2$  is sat, and  $S_1$  is sat or  $S_2$  is sat. Same reasoning if  $L^c \in M$ .

If  $L \notin M$  and  $L^c \notin M$ , there exists  $K_1 \in M$ ,  $K_2 \in M$  such that  $K_1 \in \alpha$ ,  $K_2 \in \beta$ . Hence  $S_1$  sat and  $S_2$  sat.

**(R4)**

$$L \vee \alpha \in S \quad L \vee \alpha \vee \beta \in S$$

$$S_1 = S \setminus \{L \vee \alpha \vee \beta\}$$

$S$  unsat iff  $S_1$  unsat

if

We prove the contrapositive.

$S$  sat. As  $S_1 \subsetneq S$ ,  $S_1$  is sat.

only if

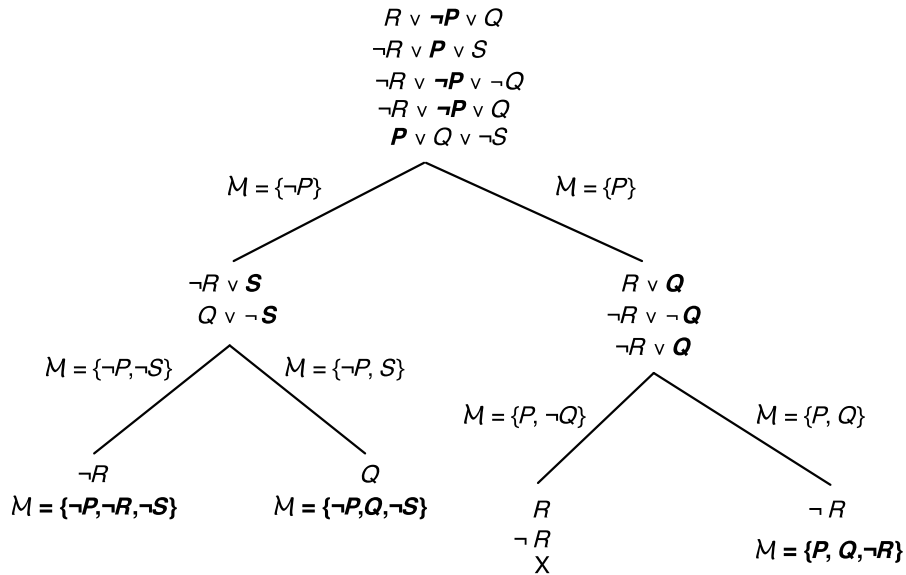
We prove the contrapositive.

$S_1$  sat, let  $M_1$  be a model of  $S_1$ . Of course,  $M_1$  is a model of  $L \vee \alpha$ ; hence (definition of a clause),  $M_1$  is also a model of  $L \vee \alpha \vee \beta$ . Therefore,  $S$  is sat.  $\square$

EXERCISE 3.31.– The answer is yes, as shown by the application of the algorithm with the strategy below.

In example 3.13, we implicitly applied the strategy “find a model as soon as possible”.

We now apply the strategy “try to find as many models as possible” (actually, in this particular case, we find all of them).



We thus found the following six models:

$$\mathcal{M}_1 = \{\neg P, Q, \neg R, \neg S\}$$

$$\mathcal{M}_2 = \{\neg P, \neg Q, \neg R, \neg S\}$$

$$\mathcal{M}_3 = \{\neg P, Q, R, S\}$$

$$\mathcal{M}_4 = \{\neg P, Q, \neg R, S\}$$

$$\mathcal{M}_5 = \{P, Q, \neg R, S\}$$

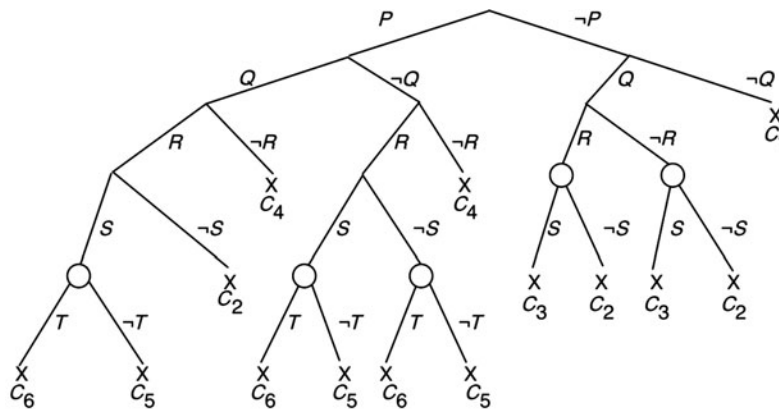
$$\mathcal{M}_6 = \{P, Q, \neg R, \neg S\}.$$

□

EXERCISE 3.32.–

a) We take the following (arbitrary) order for the basic formulas:

$$P < Q < R < S < T$$



○ : inference nodes

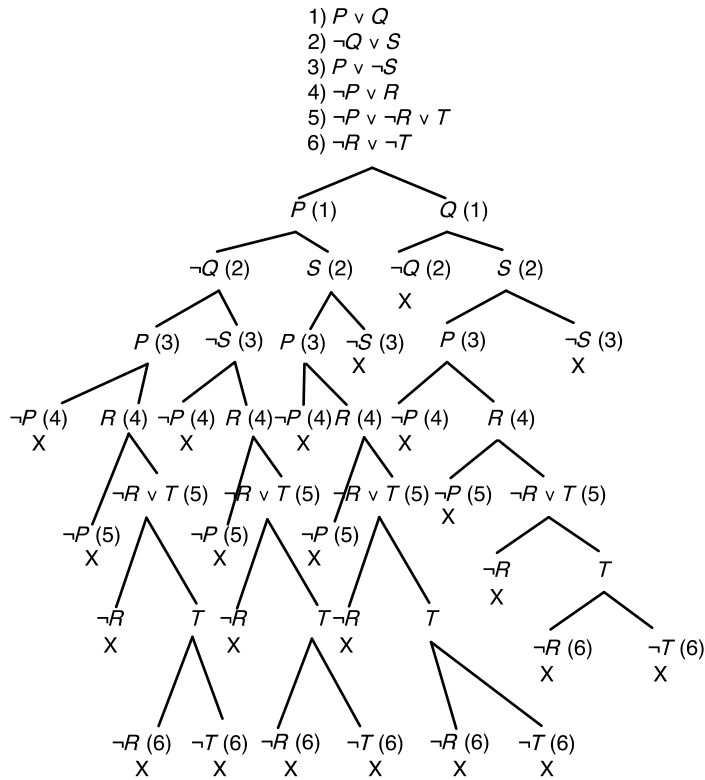
b)

The interpretation that we can give of the inference nodes is as follows:

*The interpretations denoted by the two branches going through an inference node falsify the clauses  $C_i : \alpha \vee L$  and  $C_j : \beta \vee L^c$  ( $\alpha$  and  $\beta$ : disjunctions of literals). By definition of a clause, the interpretation denoted by the branch that finishes by an inference node falsifies all the literals of  $\alpha$  and all the literals of  $\beta$ . It therefore falsifies the resolvent of  $C_i$  and  $C_j$  (see definition 3.15).*

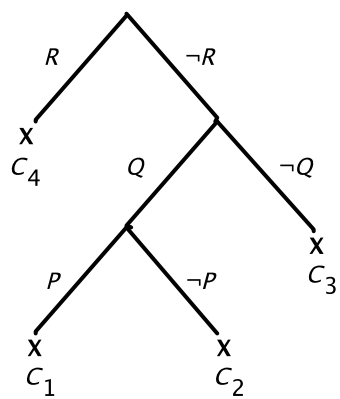


c)



□

EXERCISE 3.33.–



□

EXERCISE 3.34.– We must prove:

If  $S$  is unsat, then  $S \vdash_R \square$  (see section 3.7)

We give two proofs of this result.

PROOF 1.– We prove the result by induction on the *number of positive literals* in  $S$ .

i)  $n = 1$                       % if  $n = 0$ , then  $S$  is satisfiable.

Since  $S$  is unsatisfiable,  $S$  is necessarily of the form:

$S = \{L, L^c\}$  hence, by applying the resolution rule once we obtain:

$S \vdash_R \square$

ii)  $n > 1$

The idea is to apply a transformation that preserves the unsatisfiability, and *strictly decreases* the number of literals (to apply the induction hypothesis).

Assume  $S$  contains  $n + 1$  literals.

Choose a literal  $L \in C \in S$  that is not pure.

Consider the sets of clauses:

$$S_1 = \{C \setminus \{L\} \mid C \in S \text{ and } L^c \notin C\}$$

$$S_2 = \{C \setminus \{L^c\} \mid C \in S \text{ and } L \notin C\}$$

It was proved (see exercise 3.30) that:

$S$  is unsat 1 iff  $S_1$  is unsat and  $S_2$  is unsat.

By the induction hypothesis,  $S_1 \vdash_R \square$                       %  $S_1$  contains at most  $n$  literals.

There are two cases to consider (the second case itself leads to two cases):

a) the clauses  $C \setminus \{L\}$  do not occur in the refutation of  $S_1$ .

The clauses that occur in the refutation are also in  $S$ , hence:

$S \vdash_R \square$ .

b1) the clauses  $C \setminus \{L\}$  occur in the refutation of  $S_1$ .

In this case,  $S \vdash_R L$ , as can be proved by generalizing the following example.

- 1)  $A \vee B$
- 2)  $\neg A \vee B$
- 3)  $A \vee \neg B$
- 4)  $\neg A \vee \neg B$
- 5)  $B$  (1, 1) – (2, 1)
- 6)  $\neg B$  (3, 1) – (4, 1)
- 7)  $\square$  (5, 1) – (6, 1)

If we add a pure literal to the first clause:

- 1)  $A \vee B \vee \boxed{L}$
- 2)  $\neg A \vee B$
- 3)  $A \vee \neg B$
- 4)  $\neg A \vee \neg B$
- 5)  $B \vee \boxed{L}$  (1, 1) – (2, 1)
- 6)  $\neg B$  (3, 1) – (4, 1)
- 7)  $\boxed{L}$  (5, 1) – (6, 1)

b2) by repeating exactly the same reasoning for  $S_2$  with  $L^c$  instead of  $L$ , we conclude that:

$$S \vdash_R L^c.$$

Now for  $S$  (containing  $n + 1$  literals) we have:

$$S \vdash_R L \quad \text{and} \quad S \vdash_R L^c$$

hence, by applying once the resolution rule, we obtain:  $S \vdash_R \square$ .

PROOF 2.– We give another proof by using theorem 3.2, and the results of exercise 3.32 (b).

By theorem 3.2, if  $S$  is unsat then there exists a closed semantic tree  $T$  for  $S$ .

First note that in a closed semantic tree, there are always inference nodes, otherwise all clauses would be positive (respectively, negative), and by applying the purity principle (see exercise 3.30), we would obtain an empty set equivalent to  $S$ , which would thus be satisfiable.

By eliminating all failure nodes from  $T$ , we obtain a closed tree  $T'$  (with strictly less nodes than  $T$ ), corresponding to the set of clauses:

$$S' = \{R(C_i, C_j)\} \cup_k \underbrace{S \setminus (\{C_i\} \cup \{C_j\})}_{\text{not descendant of inference node}} ; \quad k : \text{number of inference nodes}$$

We repeat the reasoning, but this time on  $S'$ .

As we are starting with a finite closed tree and at each step we obtain strictly smaller closed trees, we necessarily obtain a finite tree with  $\neg L$  and  $L$  (of course  $L$  denotes an arbitrary literal) as left and right branches, respectively, and the corresponding inference node is  $\square$ .  $\square$

EXERCISE 3.35.– We must show:

$$\text{if } S \vdash_R \square \text{ then } S \vdash_{R+te} \square$$

te: tautological clauses elimination.

We first prove that if  $S$  is unsat then  $\mathcal{R}^n(S)$  ( $n \geq 1$ ) (see the definition of operator  $\mathcal{R}$  (definition 3.16)) is unsat.

The proof is trivial: every superset of an unsat set is unsat.

As we have proved the completeness of resolution for refutation, it suffices to apply rule R-0 of the Davis and Putnam method to  $R^n(S)$  (unsatisfiable set of clauses) for every  $n \geq 1$ .  $\square$

EXERCISE 3.36.– As resolution is *complete for refutation*, if  $S$  is unsat then  $\square$  will be derived after a finite number of steps.

Given a set of clauses  $S$ , the problem is to decide whether  $S$  is sat or unsat. As  $S$  contains a *finite* number of literals and the application of  $R_c$  does not add any new literal, after a finite number of steps, we either generate  $\square$  or we do not generate any new clause. In the latter case, we say we have *saturated* the set of clauses.

When the set of consequences of  $S$  has been saturated, the refutational completeness of  $S$  enables us to conclude that  $S$  is *satisfiable*.

An application example of this decision procedure is the detection of the satisfiability of the set of clauses (1), (2), (3) below:

- 1)  $A \vee B$
- 2)  $\neg A \vee B$
- 3)  $\neg A \vee \neg B$
- 4)  $B$  (1, 1) – (2, 1)
- 5)  $B \vee \neg B$  (1, 1) – (3, 1)
- 6)  $A \vee \neg A$  (1, 2) – (3, 2)
- 7)  $\neg A$  (2, 2) – (3, 2)

□

## EXERCISE 3.37.–

a)

- 1)  $P$
- 2)  $\neg P \vee Q$
- 3)  $\neg Q \vee R$
- 4)  $\neg Q \vee \neg R$
- 5)  $Q$  (1, 1) – (2, 1)
- 6)  $R$  (5, 1) – (3, 1)
- 7)  $\neg R$  (5, 1) – (4, 1)
- 8) □ (6, 1) – (7, 1)

b)

- 1)  $R$
- 2)  $Q \vee \neg R$
- 3)  $S \vee \neg R$
- 4)  $P \vee \neg Q \vee \neg S$
- 5)  $\neg P \vee \neg Q \vee \neg S$
- 6)  $Q$  (1, 1) – (2, 2)
- 7)  $S$  (1, 1) – (3, 2)
- 8)  $P \vee \neg S$  (6, 1) – (4, 2)
- 9)  $P$  (7, 1) – (8, 2)
- 10)  $\neg P \vee \neg S$  (6, 1) – (5, 2)
- 11)  $\neg P$  (7, 1) – (10, 2)
- 12) □ (9, 1) – (11, 1)

c)

- 1)  $P \vee Q$
- 2)  $P \vee \neg Q$
- 3)  $R \vee Q$
- 4)  $R \vee \neg Q$
- 5)  $P$  (1, 2) – (2, 2)
- 6)  $P \vee R$  (1, 2) – (4, 2)
- 7)  $R$  (3, 2) – (4, 2)

There is no way of obtaining different clauses; hence,  $S$  is *satisfiable*.

d)

We give the clausal translation of each formula:

- 1)  $\neg P$
- 2)  $\neg R \Rightarrow W$        $R \vee W$
- 3)  $Q \vee (\neg T \Rightarrow \neg P \wedge \neg S) \Leftrightarrow$   
 $Q \vee (T \vee (\neg P \wedge \neg S)) \Leftrightarrow Q \vee ((T \vee \neg P) \wedge (T \vee \neg S)) \Leftrightarrow$   
 $(Q \vee T \vee \neg P) \wedge (Q \vee T \vee \neg S)$
- 4)  $\neg P \Rightarrow (S \wedge \neg R)$   
 $P \vee (S \wedge \neg R) \Leftrightarrow (P \vee S) \wedge (P \vee \neg R)$
- 5)  $\neg Q$
- 6)  $\neg S$
- 7)  $\neg T$
- 8)  $\neg R \Rightarrow Y$        $R \vee Y$

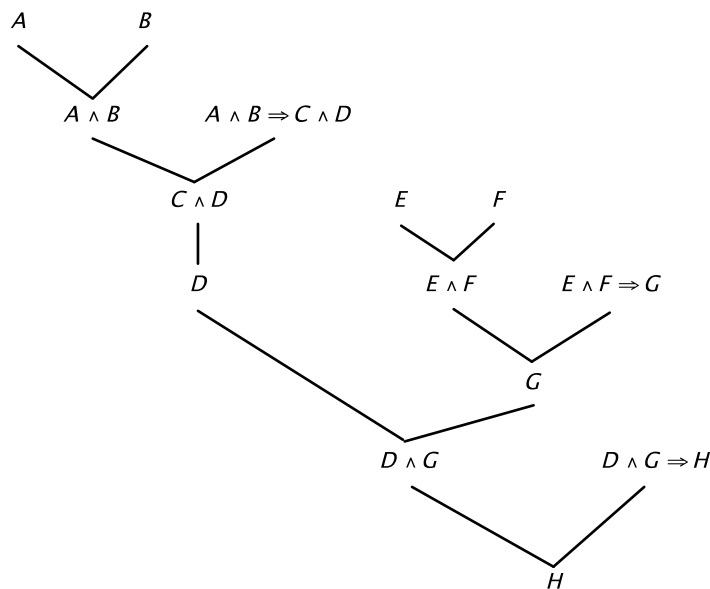
We prove that the set of clauses is unsatisfiable:

- 1)  $\neg P$
- 2)  $R \vee W$
- 3)  $Q \vee T \vee \neg P$
- 4)  $Q \vee T \vee \neg S$
- 5)  $P \vee S$
- 6)  $P \vee \neg R$

- 7)  $\neg Q$
- 8)  $\neg S$
- 9)  $\neg T$
- 10)  $R \vee Y$
- 11)  $P$  (5, 2) – (8, 1)
- 12)  $\square$  (11, 1) – (1, 1)

e)

i) If we did not know the resolution rule, we would probably give a proof similar to the following:



ii) By applying the resolution rule. We first translate into clausal form:

$$\neg(A \wedge B) \vee (C \wedge D) \longrightarrow (\neg A \vee \neg B) \vee (C \wedge D) \longrightarrow (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg B \vee D)$$

$$E \wedge F \Rightarrow G \longrightarrow \neg(E \wedge F) \vee G \longrightarrow \neg E \vee \neg F \vee G$$

$$G \wedge D \Rightarrow H \longrightarrow \neg(G \wedge D) \vee H \longrightarrow (\neg G \vee \neg D) \vee H \longrightarrow \neg G \vee \neg D \vee H$$

By negating the conclusion, we obtain the set of clauses 1–9 below and a refutation using resolution.

- 1)  $\neg A \vee \neg B \vee C$
- 2)  $\neg A \vee \neg B \vee D$
- 3)  $\neg E \vee \neg F \vee G$
- 4)  $\neg G \vee \neg D \vee H$
- 5)  $A$
- 6)  $B$
- 7)  $F$
- 8)  $E$
- 9)  $\neg H$
- 10)  $D$  (2), (5), (6)
- 11)  $G$  (3), (7), (8)
- 12)  $H$  (4), (10), (11)
- 13)  $\square$  (9), (12) □

REMARK 12.8.— In (10), (11), and (12) we used the *hyperresolution* rule (we “compress” several resolution steps into only one). □

f) We must consider the set of clauses 1–8 below, obtain all possible resolvents and reach saturation without generating  $\square$ .

- 1)  $\neg S \vee \neg M \vee T$
- 2)  $M \vee S \vee C$
- 3)  $\neg S \vee M \vee \neg A$
- 4)  $C \vee \neg T$
- 5)  $A$
- 6)  $\neg T \vee \neg B$
- 7)  $\neg C \vee \neg B$
- 8)  $B$  □

EXERCISE 3.38.— Not necessarily, the set of clauses:

$$\{A, \neg A, B, \neg B\}$$

does not contain any pure literal, but contains proper subsets that are unsatisfiable. □



## EXERCISE 3.39.–

a)

- 1)  $R$
- 2)  $\neg R \vee Q$
- 3)  $\neg R \vee S$
- 4)  $\neg P \vee \neg Q \vee \neg S$
- 5)  $P \vee \neg Q \vee \neg S$
- 6)  $\neg Q \vee \neg S$     (4, 1) – (5, 1)
- 7)  $\neg R \vee \neg S$     (6, 1) – (2, 2)
- 8)  $\neg S$             (7, 1) – (1, 1)
- 9)  $\neg R$             (8, 1) – (3, 2)
- 10)  $\square$             (9, 1) – (1, 1)

b) No. Take for example the *unsatisfiable* set of clauses:

$$S = \{A \vee B, \neg A \vee B, A \vee \neg B, \neg A \vee \neg B\}$$

We cannot deduce  $\square$  using an input strategy, because the set does not contain any unit clause. Indeed,  $\square$  can only be generated by two complementary unit clauses, one of which must belong to  $S$  (this is required by the input strategy).  $\square$

EXERCISE 3.40.– No. Take the same unsatisfiable set as in exercise 3.39:

$$S = \{A \vee B, \neg A \vee B, A \vee \neg B, \neg A \vee \neg B\}$$

We cannot apply the resolution rule with a unit strategy.  $\square$ EXERCISE 3.41.– Every interpretation  $I$  of  $S$  can be represented (see example 3.4, point 2.) by:

$I = \{L_1^*, L_2^*, \dots, L_n^*\}$ , where  $L_i^* = L_i$  or  $L_i^* = L_i^c$ ; ( $1 \leq i \leq n$ ) and  $L_i \in \text{Propset}(S)$  (see definition 3.8).

As  $S$  contains all the clauses of length  $n$  that can be formed with  $n$  propositional symbols,  $S$  contains a clause:

$C_j : L_{j1} \vee L_{j2} \vee \dots \vee L_{jn}$  where:

$$L_{jk} = (L_i^*)^c \quad (1 \leq k \leq n)$$

In other words, if  $L_i$  is in  $I$  then  $\neg L_i$  is in  $C_j$  and if  $\neg L_i$  is in  $I$  then  $L_i$  is in  $C_j$ .

$C_j$  is falsified by  $I$ ; hence,  $S$  is falsified by  $I$ .

This reasoning holds for all interpretations of  $S$ , which is therefore unsatisfiable.  $\square$

EXERCISE 3.42.–

a) Every interpretation  $I$  of  $S$  contains (at least)  $p$  literals  $L_1, L_2, \dots, L_p$  with the same sign.

By construction,  $S$  contains a clause  $C: L_1^c, L_2^c, \dots, L_p^c$ , which will of course be evaluated to **F** in  $I$ .

Therefore,  $S$  is unsatisfiable.

b) No.

Take as a counter example  $n = 2$  (say  $\{A, B\}$ ); hence,  $p = 2$

$S = \{A \vee B, \neg A \vee \neg B\}$ , which is *satisfiable* (models  $\{A, \neg B\}, \{\neg A, B\}$ )

(We could have chosen  $n = 4, n = 6, \dots$ )

The explanation of the counter examples is that there could be, with the new hypothesis on  $p$ , interpretations containing only  $q$  literals of the same sign, with  $q < p$ .  $\square$

EXERCISE 3.43.–

a)

We consider the propositions  $P_{ij}$  ( $1 \leq i \leq n, 1 \leq j \leq n - 1$ ) meaning  $\varphi(i) = j$ .

We give the set of clauses specifying that such an injective function exists. This set must therefore be *unsatisfiable*.

The clauses specifying all possible images of the elements of the domain of cardinality  $n$  ( $S_n$ ):

$$P_{i1} \vee P_{i2} \vee \dots \vee P_{i(n-1)} \quad (1 \leq i \leq n) (*)$$

The clauses specifying that the function is injective will be of the form:

$\neg(P_{ik} \wedge P_{jk})$ , i.e.:

$$\neg P_{ik} \vee \neg P_{jk} \quad (1 \leq i < j \leq n, \quad 1 \leq k \leq n-1) \quad (* *)$$

We will thus have:

$n \times (n-1)$  propositional symbols

$n$  clauses of length  $n-1$  (\*)

$n \times (n-1)^2 \div 2$  clauses of length 2 (\* \*)

The general case:

$$\bigwedge_{i=1}^n \bigvee_{j=1}^{n-1} P_{ij} \wedge \bigwedge_{k=1; i \neq j}^{n-1} (\neg P_{ik} \vee \neg P_{jk})$$

b)  $S_3$  (clauses 1–9):

- 1)  $P_{11} \vee P_{12}$
- 2)  $P_{21} \vee P_{22}$
- 3)  $P_{31} \vee P_{32}$
- 4)  $\neg P_{11} \vee \neg P_{21}$
- 5)  $\neg P_{21} \vee \neg P_{31}$
- 6)  $\neg P_{11} \vee \neg P_{31}$
- 7)  $\neg P_{12} \vee \neg P_{22}$
- 8)  $\neg P_{22} \vee \neg P_{32}$
- 9)  $\neg P_{12} \vee \neg P_{32}$
- 10)  $P_{12} \vee \neg P_{21} \quad (1, 1) - (4, 1)$
- 11)  $\neg P_{21} \vee \neg P_{32} \quad (10, 1) - (9, 1)$
- 12)  $P_{22} \vee \neg P_{32} \quad (11, 1) - (2, 1)$
- 13)  $\neg P_{32} \quad (12, 1) - (8, 1)$
- 14)  $P_{31} \quad (13, 1) - (3, 2)$
- $\vdots$

REMARK 12.9.– It is possible to prove that the minimal number of clauses necessary to refute the set of clauses specifying the pigeonhole  $S_n$  is:

$$(n-1) \times (n+2) \times 2^{(n-3)}$$

For  $S_3$ , the shortest refutation then contains  $2 \times 5 \times 1 = 10$  clauses.

The pigeonhole was used to prove that the resolution method is *exponential*, i.e. that there exists at least a set of clauses for which the shortest refutation is exponential in the number of clauses.  $\square$

REMARK 12.10.– We sometimes specify the pigeonhole problem by translating: “If we define a function from  $n + 1$  objects onto  $n$  objects then there exists (at least) two objects of the domain with the same image”, which translates into the *tautological* schema:

$$\bigwedge_{i=1}^{n+1} \bigvee_{j=1}^n P_{ij} \Rightarrow \bigvee_{k=1}^n \bigvee_{i=1}^{n+1} \bigvee_{j=i+1}^{n+1} (P_{ik} \wedge P_{jk}) \quad \square$$

EXERCISE 3.44.–

No. We give two proofs of this result.

a)

As all clauses are of the form:

$$\bigvee_i P_i \vee \bigvee_j N_j \quad (1 \leq i, 1 \leq j)$$

with:

$P_i$ : positive literal;

$N_j$ : negative literal;

the interpretations  $I = \{P_i\}$  and  $J = \{N_j\}$  are models of the set of clauses that can therefore not be unsatisfiable.

b)

As the resolution is complete for refutation, we can assume without loss of generality that we are trying to detect the unsatisfiability of the set of clauses using the resolution rule.

The set does not contain any unit clause. Indeed, as a unit clause contains only one literal and as a literal is either positive or negative, if the set contained a unit clause, the hypotheses would be violated.

Thus, all clauses are of the form:

$$\neg A_1 \vee \dots \vee \neg A_m \vee B_1 \vee \dots \vee B_n \quad (m \geq 1, n \geq 1)$$

Without loss of generality, we reason on clauses of length 2. The application of the resolution rule to

$$\neg L \vee P \quad (\text{if } L \text{ and } P \text{ are the same literal, then the clause can be eliminated})$$

$$L \vee \neg Q$$

produces a clause of the form:

$$P \vee \neg Q$$

Therefore, we can never obtain the complementary unit clauses that are necessary to obtain  $\square$ .

The required corollary is trivial to obtain. The question is whether a set of clauses containing a positive clause (respectively, negative clause) but no negative clause (respectively, positive clause) can be unsatisfiable. The answer is no. In the first case, the interpretation

$$I = \{P_i\}$$

is a model. In the second case, the interpretation

$$J = \{N_j\}$$

is a model.

Hence the conclusion.  $\square$

EXERCISE 3.45.– We use the propositional symbol:

$P_i^j$ : country  $i$  is colored with color  $j$ .

Each country is colored with exactly one color:

$$\text{for } 1 \leq i \leq N: (P_i^1 \wedge \neg P_i^2 \wedge \neg P_i^3) \vee (P_i^2 \wedge \neg P_i^1 \wedge \neg P_i^3) \vee (P_i^3 \wedge \neg P_i^1 \wedge \neg P_i^2)$$

By transformation into clausal form (simplifying the clauses containing tautologies and applying the subsumption rule), we obtain the equivalent specification:

$$\text{(Sch1) for } 1 \leq i \leq N: (P_i^1 \vee P_i^2 \vee P_i^3) \wedge (\neg P_i^1 \vee \neg P_i^2) \wedge (\neg P_i^1 \vee \neg P_i^3) \wedge (\neg P_i^2 \vee \neg P_i^3)$$

(Sch1) could also have been obtained by translating: “country  $i$  must be colored with a color (first conjunct) and we cannot color it with more than one color (three other conjuncts)”.

The complexity of the specification of these clauses is less than  $3 \times N$ , or in other words,  $O(N)$ .

Two neighboring countries cannot be colored the same way:

(Sch2) for all neighboring countries  $i, j$ :  $\neg((P_i^1 \wedge P_j^1) \vee (P_i^2 \wedge P_j^2) \vee (P_i^3 \wedge P_j^3))$

Every country has at most  $N - 1$  neighbors, hence there are less than  $N^2$  clauses similar to the clause above. Hence, the complexity of the specification of this second part is  $O(N^2)$ .

The global complexity is thus  $O(N^2)$ . □

REMARK 12.11.– In the coloring problem of example 9.19, the solution was obtained using a specification with constraints.

We could apply the formula schemata (Sch1) and (Sch2) above to specify the same map and the method of semantic tableaux (see section 3.6) to find all the models (i.e. the solutions to the problem). Of course, the solutions must be the same. It could be interesting to compare both approaches (one numerical, the other logical). □

EXERCISE 3.46.–

a)

We assume that the models are represented by the set of propositional symbols that are evaluated to **T** in the models (form 3 in example 3.4).

Of course, the theorem does not depend on the adopted representation of models.

We note:

$$H = \{C_1, C_2, \dots, C_n\}$$

$Mod(H)$ : the set of models of  $H$

$$\mathcal{N} \subseteq Mod(H) (\mathcal{N} \neq \emptyset)$$

$$M_{\cap} = \cap M_k, M_k \in \mathcal{N}$$

PROOF.– To derive a contradiction, we assume:

$$\not\models_{M_{\cap}} H, \text{ hence, for at least one clause } C_j \in H :$$

$$\not\models_{M_{\cap}} C_j$$

There are two cases to consider (depending on the different forms of Horn clauses, see definition 3.21).

i)  $C_j: A \vee \neg B_1 \vee \neg B_2 \vee \dots \vee \neg B_m; m \geq 0$  ( $m = 0$  corresponds to a unit positive clause)

To evaluate a clause to **F**, it is necessary to evaluate all of its literals to **F** (see definition 3.14), i.e.:

$$A \notin M_\cap \text{ and } B_1, B_2, \dots, B_m \in M_\cap$$

hence, by definition of an intersection, there exists  $M_k \in \mathcal{N}$  such that:

$$A \notin M_k \text{ and } B_1, B_2, \dots, B_m \in M_k$$

but then  $M_k$  is not a model of  $H$  (as it falsifies one of its clauses:  $C_j$ ). *Contradiction.*

ii)  $C_j: \neg B_1 \vee \neg B_2 \vee \dots \vee \neg B_m; m > 0$

To evaluate  $C_j$  to **F**:

$$B_1, B_2, \dots, B_m \in M_\cap$$

hence, by definition of an intersection, for all  $M_k \in \mathcal{N}$ :

$$B_1, B_2, \dots, B_m \in M_k$$

but then the  $M_k$ 's are not models of  $H$  (as they falsify one of its clauses:  $C_j$ ). *Contradiction.*

b) No. Take  $H = \{A \vee B\}$

$$\text{Mod}(H) = \{\{A\}, \{B\}, \{A \vee B\}\}$$

$$\text{and } \mathcal{N} = \{\{A\}, \{B\}\}$$

then  $M_\cap = \emptyset$ , which is not a model of  $H$ . □

EXERCISE 4.1.–

a)

$$\Gamma_0 = \{f(x, g(x, y)) \doteq f(g(y, z), g(g(h(u), y), h(u)))\}$$

R-4 on  $\Gamma_0 =$  yields:

$$\Gamma_1 = \{x \doteq g(y, z), g(x, y) \doteq g(g(h(u), y), h(u))\}$$

R-4 on  $\Gamma_1 =$  yields:

$$\Gamma_2 = \{x \doteq g(y, z), x \doteq g(h(u), y), y \doteq h(u)\}$$

R-3 on  $\Gamma_2 =$  yields:

$$\Gamma_3 = \{x \doteq g(y, z), g(y, z) \doteq g(h(u), y), y \doteq h(u)\}$$

R-4 on  $\Gamma_3 =$  yields:

$$\Gamma_4 = \{x \doteq g(y, z), y \doteq h(u), z \doteq y\}$$

R-5 on  $\Gamma_4 =$  yields:

$$\Gamma_5 = \{x \doteq g(h(u), h(u)), y \doteq h(u), z \doteq h(u)\}$$

There are no rules that can be applied, we halt.

$$\text{Solution: } \sigma = \{x \leftarrow g(h(u), h(u)), y \leftarrow h(u), z \leftarrow h(u)\}$$

b)

$$f(x, f(u, x)) \doteq f(f(y, a), f(z, f(b, z)))$$

- |                             |                 |
|-----------------------------|-----------------|
| 1) $x \doteq f(y, a)$       | R - 4           |
| 2) $u \doteq z$             | R - 4 two times |
| 3) $x \doteq f(b, z)$       | R - 4 two times |
| 4) $f(y, a) \doteq f(b, z)$ | (1), (3), R - 3 |
| 5) $y \doteq b$             | (4), R - 4      |
| 6) $z \doteq a$             | (4), R - 4      |
| 7) $u \doteq a$             | (2), (6), R - 5 |

$$\text{Solution: } \sigma = \{x \leftarrow f(b, a), y \leftarrow b, u \leftarrow a, z \leftarrow a\}$$

c)

$\theta$	$\sigma$	$\sigma \circ \theta(\theta\sigma)$
$x \mapsto a$	$x \mapsto b$	$x \mapsto a$
$y \mapsto f(z)$	$y \mapsto y$	$y \mapsto f(c)$



$$\begin{array}{lll}
 z \mapsto x & z \mapsto c & z \mapsto b \\
 u \mapsto u & u \mapsto d & u \mapsto d \\
 v \mapsto v & v \mapsto v & v \mapsto v \\
 w \mapsto w & w \mapsto w & w \mapsto w \\
 \vdots & \vdots & \vdots
 \end{array}$$

$$\sigma \circ \theta = \{x \leftarrow a, y \leftarrow f(c), z \leftarrow b, u \leftarrow d\}$$

d)

Yes, it is always necessary. Consider the equation:

$$f(x, g(x)) \doteq f(h(y), y)$$

$$\text{Clearly, } \text{Var}(f(x, g(x))) \cap \text{Var}(f(h(y), y)) = \emptyset$$

If we do not apply the cycle rule, we have:

$$x \doteq h(y)$$

$$y \doteq g(x)$$

i.e.:

$$x = h(g(x))$$

which is an infinite term.

A sufficient condition to be able to deal without the cycle rule is:

$\text{Var}(t_1) \cap \text{Var}(t_2) = \emptyset$  and  $(t_1$  or  $t_2)$  linear (a term  $t$  is linear iff all of its variables occurs *at most once* in  $t$ ).  $\square$

EXERCISE 4.2.– The possibilities of identification of the (sub)formulas corresponding to  $A$ ,  $B$ ,  $C$  in  $CD$  are:

1)

$$e(\underbrace{X}_A, \underbrace{e(X, e(Y, Y))}_B)$$

$$\underbrace{e(Z, Z)}_C$$

2)

$$e(\underbrace{X}_B, \underbrace{e(X, e(Y, Y))}_A)$$

$$\underbrace{e(Z, Z)}_C$$

3)

$$\underbrace{e(X, e(X, e(Y, Y)))}_C$$

$$e(\underbrace{Z}_A, \underbrace{Z}_B)$$

4)

$$\underbrace{e(X, e(X, e(Y, Y)))}_C$$

$$e(\underbrace{Z}_B, \underbrace{Z}_A)$$

There remains to compute three mgus and three direct consequences in the case in which the mgus exist, cases (3) and (4) are of course identical.

$$\sigma = \{X \leftarrow e(Z, Z)\}$$

$$\sigma B = e(e(Z, Z), e(Y, Y))$$

$$\sigma = \{X \leftarrow e(Y, Y)\}$$

$$\sigma B = e(Y, Y)$$

$$\sigma = \{Z \leftarrow e(X, e(X, e(Y, Y)))\}$$

$$\sigma B = e(X, e(X, e(Y, Y)))$$

We note that the only useful direct consequence is the first consequence (the others are wffs that coincide with the premises).  $\square$

## EXERCISE 4.4.–

– To be able to compare the rules and verify what is asked, we need to choose a same language to represent the three rules. We choose the language of clauses.

– To be able to apply the UNIFICATION algorithm (as well as the modified UNIFICATION), we consider connectives as functional symbols.

– Terms are formed from functional symbols of fixed arities  $\vee^{(2)}$  and  $\neg^{(1)}$ , which explains why we use  $\epsilon$ , which ordinarily denotes the empty string of symbols (i.e. the disjunction without any disjunct) in the terms representing the inference rules.

– this is a *matching* problem; hence, we use for  $R$  symbols that ordinarily denote variables.

–  $A, B, X$  denote literals;  $\epsilon, \mathcal{A}, \mathcal{B}, \mathcal{X}, \mathcal{Y}$  denote clauses.

$MP$ :  $\text{concl}(\text{and}(\vee(A, \epsilon), \vee(\neg A, \mathcal{B})), \mathcal{B})$

$MT$ :  $\text{concl}(\text{and}(\vee(\neg B, \epsilon), \vee(\neg \mathcal{A}, \mathcal{B})), \neg \mathcal{A})$

$R$ :  $\text{concl}(\text{and}(\vee(X, \mathcal{X}), \vee(\neg X, \mathcal{Y})), \vee(\mathcal{X}, \mathcal{Y}))$

The solution of equation:

$\text{concl}(\text{and}(\vee(A, \epsilon), \vee(\neg A, \mathcal{B})), \mathcal{B}) \doteq \text{concl}(\text{and}(\vee(X, \mathcal{X}), \vee(\neg X, \mathcal{Y})), \vee(\mathcal{X}, \mathcal{Y}))$

found by UNIFICATION is:

$\sigma_{R-MP} = \{X \leftarrow A, \mathcal{X} \leftarrow \epsilon, \mathcal{Y} \leftarrow \mathcal{B}\}$

The solution of equation:

$\text{concl}(\text{and}(\vee(\neg B, \epsilon), \vee(\neg \mathcal{A}, \mathcal{B})), \mathcal{A}) \doteq \text{concl}(\text{and}(\vee(X, \mathcal{X}), \vee(\neg X, \mathcal{Y})), \vee(\mathcal{X}, \mathcal{Y}))$

found by modified UNIFICATION is:

$\sigma_{R-MT} = \{X \leftarrow B, \mathcal{X} \leftarrow \mathcal{A}, \mathcal{Y} \leftarrow \epsilon\}$

□

## EXERCISE 5.1.–

a)

$D = \mathbb{N}$

$a \mapsto 0$

$f(x) \mapsto \text{succ}(x)$       %  $\text{succ}(x)$ : the successor of  $x$  (i.e.  $x + 1$ )

$$g(x, y) \mapsto x + y$$

$$P(x, y) \mapsto x > y$$

b)

Model:

$$D = \mathbb{Z}$$

$$a \mapsto 0$$

$$f(x, y) \mapsto x - y$$

$$P(x, y) \mapsto x > y$$

Counter-model:

Only change

$$a \mapsto -5$$

c)

$$D = \mathbb{N}$$

$$f(x, y) \mapsto x + y$$

$$P(x, y) \mapsto x > y$$

d)

$$D = \mathbb{N}$$

$$f(x, y) \mapsto x \times y$$

$$P(x, y) \mapsto x \geq y \quad \% y = 0$$

e)

$$D = \mathbb{N}$$

$$a \mapsto 0$$

$$f(x, y) \mapsto x + y$$

$$P(x, y) \mapsto x = y$$

f)

$$D = \mathbb{N}$$

$$P(x, y) \mapsto x < y$$

g)

Model:

$$D = \mathbb{Q}$$

$$P(x, y) \mapsto x < y$$

$$Q(x, y) \mapsto x = y$$

Counter-model:

Change  $D = \mathbb{N}$ 

h) No.

Take, for example:

$$D = \mathbb{N}$$

$$P(x) \mapsto x \text{ is even}$$

$$Q(x) \mapsto x \text{ is odd.}$$

i)

$$P(x, y) \mapsto x = y$$

□

REMARK 12.12.– One characteristic of the equality relation is that it can be applied to (and have a meaning in) any domain of discourse. □

j)

 $j_1)$ 

$$D = \mathbb{R}$$

$P(x, y) \mapsto x$  and  $y$  have the same sign

$j_2)$

$D = \{1, 2\}$

$P(x, y) \mapsto x = y$

$j_3)$

Counter-model with:

$D = \mathbb{N}$

$P(x, y) \mapsto x = y$

(take, for example,  $x = 3, y = 4$  and  $z = 5$ )

k)

$k_1)$

This formula does not have any finite model. It is a *total, transitive* and *irreflexive* relation. See also example 9.33.

We prove this by *reductio ad absurdum*. Assume it admits a finite model of domain:

$D = \{a_1, a_2, \dots, a_n\} \ n \in \mathbb{N}$

$P \mapsto \mathfrak{P}$

As  $\mathfrak{P}$  is total, for any  $a_i \in D$  :

$\mathfrak{P}(a_i, a_j)$  and  $a_i \neq a_j$  ( $\mathfrak{P}$  is irreflexive)

also,

$\mathfrak{P}(a_j, a_k)$  and  $a_j \neq a_k$  and  $\mathfrak{P}(a_i, a_k)$  ( $\mathfrak{P}$  is transitive)

also,

$\mathfrak{P}(a_k, a_l)$  and  $a_k \neq a_l$  and  $\mathfrak{P}(a_i, a_l)$  and  $\mathfrak{P}(a_j, a_l)$

...

but as  $D$  is finite (and of cardinality  $n$ ), after at most  $n - 1$  steps, we must produce (by transitivity)  $\mathfrak{P}(a_q, a_r)$  with  $q = r$  (because  $a_q$  is related to one of the  $a_i$ s in  $D$ , which will be reached sooner or later).

This is contradictory because  $\mathfrak{P}$  is irreflexive.

Therefore, this formula has no finite model.

$k_2$ )

$$D = \mathbb{R}$$

$$P(x, y) \mapsto x < y$$

$k_3$ )

$$D = \mathbb{N}$$

$$P(x, y) \mapsto x < y$$

1)

We can give infinitely many models for this formula by choosing:

$$D \neq \emptyset$$

$$E(x, y) \mapsto x = y$$

$$A(x, y) \mapsto x \text{ and } y \text{ are in relation } A.$$

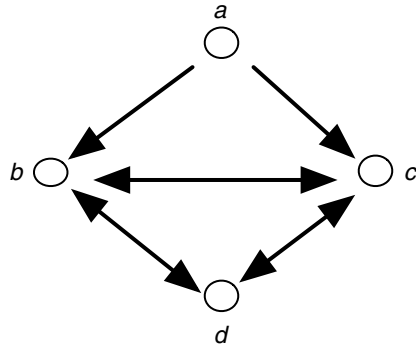
For relation  $A$ , we choose any relation such that each element is in relation with *exactly two other* elements.

For example:

$$D = \{a, b, c, d\}$$

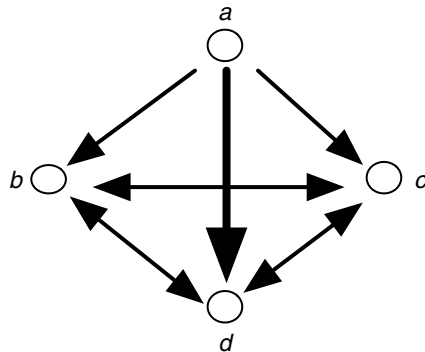
$$A^M = \{(a, b), (a, c), (b, c), (b, d), (c, b), (c, d), (d, b), (d, c)\}$$

Which can be represented by the following graph:



We obtain a counter-model by adding at least a couple (an edge) to the relation (to the graph representing the relation), for example:

$$A^M = \{(a, b), (a, c), (\mathbf{a}, \mathbf{d}), (b, c), (b, d), (c, b), (c, d), (d, b), (d, c)\}$$



m)

$m_1$ )

$D = \mathbb{N}$

$f \mapsto \mathfrak{f}$

with

$$\mathfrak{f}(2 \times x) = x$$

$$\mathfrak{f}(2 \times x + 1) = x$$



meaning that for each  $x$ , the corresponding  $y$  and  $z$  are  $y = 2 \times x$  and  $z = 2 \times x + 1$

The values of  $y$  and  $z$  can of course be interchanged.

$m_2$ )

This formula has no finite model. We assume a finite domain exists, with domain of discourse:

$$D = \{a_1, a_2, \dots, a_n\} \quad n \in \mathbb{N}$$

and derive a contradiction. By definition of a model, for all  $k \in D$  there exists  $i, j, \quad i \neq j$  such that:

$$f(a_i) = k \text{ and } f(a_j) = k$$

Since  $f$  is not injective and  $D$  is finite:

$$\text{range}(f) \subsetneq \text{domain}(f)$$

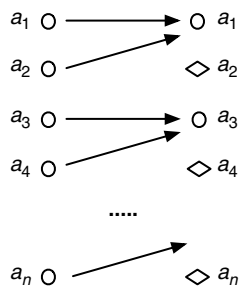
To satisfy the formula (see definition 5.6), all elements of:

$\text{domain}(f) \setminus \text{range}(f)$  must have antecedents in  $D = \text{domain}(f)$ , but these potential antecedents already have values in the range ( $f$ ).

It is impossible (definition of a function) to assign other values to them. The finite model hypothesis leads to a *contradiction*.

Therefore, this formula does not have any finite model.

A way of visualizing this reasoning is to draw the graph of function  $f$ . The elements marked with  $\diamond$  correspond to those that cannot be reached in a finite domain of discourse.



n) No.

$$\underbrace{\forall x \forall y ((f(x) = f(y)) \Rightarrow (x = y))}_{f \text{ injective}} \wedge \underbrace{\exists x \forall y f(y) \neq x}_{f \text{ not onto}}$$

See example 9.31, the definition of a finite set:

The set  $E$  is finite iff all injective functions  $E \rightarrow E$  are also onto.

See also example 9.32.

o)

Consider the interpretation  $\mathcal{I}$ :

$$D = \{0, a\}$$

$$s \mapsto \mathfrak{s}$$

$$\mathfrak{s}(0) = 0$$

$$\mathfrak{s}(a) = a$$

$$+ \mapsto \mathfrak{+}$$

$$\forall u \forall v . u \mathfrak{=} v = v$$

$$= \mapsto = \quad \% = \text{ has the same meaning as in mathematics.}$$

We verify that  $\mathcal{I}$  is a model of (1) and (2) and a counter-model of (3):

1)

$$0 \mathfrak{=} 0 \quad \mathbf{T} \quad 0 \mathfrak{=} a = a = a \quad \mathbf{T}$$

2)

$$\mathfrak{s}(0) \mathfrak{=} 0 = \mathfrak{s}(0 \mathfrak{=} 0)$$

$$0 = \mathfrak{s}(0)$$

$$0 = 0 \quad \mathbf{T}$$

$$\mathfrak{s}(0) \mathfrak{=} a = \mathfrak{s}(0 \mathfrak{=} a)$$

$$0 \mathfrak{=} a = \mathfrak{s}(a)$$

$$a = a \quad \mathbf{T}$$

$$\mathfrak{s}(a) \neq 0 = \mathfrak{s}(a \neq 0)$$

$$a \neq 0 = \mathfrak{s}(0)$$

$$0 = 0 \quad \mathbf{T}$$

$$\mathfrak{s}(a) \neq a = \mathfrak{s}(a \neq a)$$

$$a \neq a = \mathfrak{s}(a \neq a)$$

$$a = \mathfrak{s}(a)$$

$$a = a \quad \mathbf{T}$$

3)

$$a \neq 0 = a$$

$$0 = a \quad \mathbf{F}$$

p)

$$D = \mathbb{N}$$

$$a \mapsto 0$$

$$s \mapsto \mathbf{succ}$$

$$p \mapsto \mathbf{pred}$$

$$f \mapsto +$$

□

EXERCISE 5.2.– We use the same idea as in example 5.13, i.e. we consider a formula of the form  $\mathcal{A} \Rightarrow \forall x P(x)$  and we focus on the case in which  $n \geq 2$ . The problem is thus to find an adequate wff  $\mathcal{A}$ .

There exist sets of arbitrary finite cardinalities (think for example of the finite subsets of  $\mathbb{N}$ ). To specify this, we must say that:

- 1) there exist  $n$  objects;
- 2) these objects are all different (so that the set is indeed of cardinality  $n$ ).

1) translates into:  $\exists x_1 \exists x_2 \dots \exists x_n. P(x_1) \wedge P(x_2) \wedge \dots \wedge P(x_n)$

Meaning that there are  $n$  objects with some property  $P$ ; this is the usual method of defining sets.

2) To specify that they are different, it suffices to state that if an object  $x_i$  has a property, then another object  $x_j$  ( $i \neq j$ ) does not have it. Thus, it cannot be the same object.

The only remaining problem is how to choose (name) these properties. If we choose  $P$  (the same as the one used to define the set), the wff  $\mathcal{A}$  would be contradictory and the implication  $\mathcal{A} \Rightarrow \forall x P(x)$  would be trivially **T**; similarly, if we choose only one property distinct from  $P$  (simple to verify with  $n = 2$  and  $n = 3$ ).

The solution is simple: it suffices to choose distinct properties, i.e.:

$$\mathcal{A} : \exists x_1 \exists x_2 \dots \exists x_n. P(x_1) \wedge P(x_2) \wedge \dots \wedge P(x_n) \wedge$$

$$Q_1(x_1) \Leftrightarrow \neg Q_1(x_2) \wedge$$

$$Q_2(x_1) \Leftrightarrow \neg Q_2(x_3) \wedge$$

$$\vdots$$

$$Q_n(x_1) \Leftrightarrow \neg Q_n(x_n) \wedge$$

$$Q_{n+1}(x_2) \Leftrightarrow \neg Q_{n+1}(x_3) \wedge$$

$$\vdots$$

$$Q_N(x_{n-1}) \Leftrightarrow \neg Q_N(x_n)$$

with  $N = \frac{n \times (n-1)}{2}$  (i.e. all possible combinations of two objects among  $n$ ).

There remains to prove (by induction) that the proposed formula is  $n$ -valid for all  $n \in \mathbb{N}$  ( $n \geq 2$ ).

–  $n = 2$ . It is very simple to verify (for example, using the method of semantic tableaux) that the formula  $[ \exists x_1 \exists x_2. P(x_1) \wedge P(x_2) \wedge Q(x_1) \Leftrightarrow Q(x_2) ] \Rightarrow [ \forall x P(x) ]$  is  $n$ -valid (i.e. 2-valid).

–  $n = N + 1$

We must add  $\exists P(x_{n+1})$  together with the  $n$  (i.e.  $\frac{(n+1) \times n}{2} - \frac{n \times (n-1)}{2}$ ) formulas:

$$Q_{N+1}(x_1) \Leftrightarrow \neg Q_{N+1}(x_{n+1})$$

$$Q_{N+2}(x_2) \Leftrightarrow \neg Q_{N+2}(x_{n+1})$$

$$\vdots$$

$$Q_{N+n}(x_n) \Leftrightarrow \neg Q_{N+n}(x_{n+1})$$

In branch (1) of the tree, there would be  $\neg P(x_0)$  (i.e.  $\neg \forall x P(x) \rightarrow \exists x \neg P(x)$

$\rightarrow \neg P(x_0)(x \leftarrow x_0)$ ) together with  $P(x_1), P(x_2), \dots, P(x_{n+1})$ , i.e.  $n + 2$  potential constants. To work in universes of cardinality  $n + 1$  and using the induction hypothesis, we shall let  $x_i = x_{n+1}$  ( $1 \leq i \leq n$ ). The branches of depth  $N + i$  ( $1 \leq i \leq n$ ) will all contain either  $Q_{N+i}(x_i)$  and  $\neg Q_{N+i}(x_i)$  or  $\neg Q_{N+i}(x_i)$  and  $Q_{N+i}(x_i)$  (of course, this is done under the assumption, which does not incur any loss of generality, that the tree was developed in increasing order of the variable indices) and whatever the chosen instantiation, will all be closed.

The formulas for (unbounded) arbitrary cardinalities can be obtained using the following *schema*:

$$\bigwedge_{i=1}^{n \geq 2} P(a_i) \bigwedge_{j=1}^{n-1} \bigwedge_{k=2; k > j}^n (Q_l(a_j) \Leftrightarrow \neg Q_l(a_k))$$

thus using:

$$N = \frac{n \times (n-1)}{2}$$

unary predicates  $Q_l$

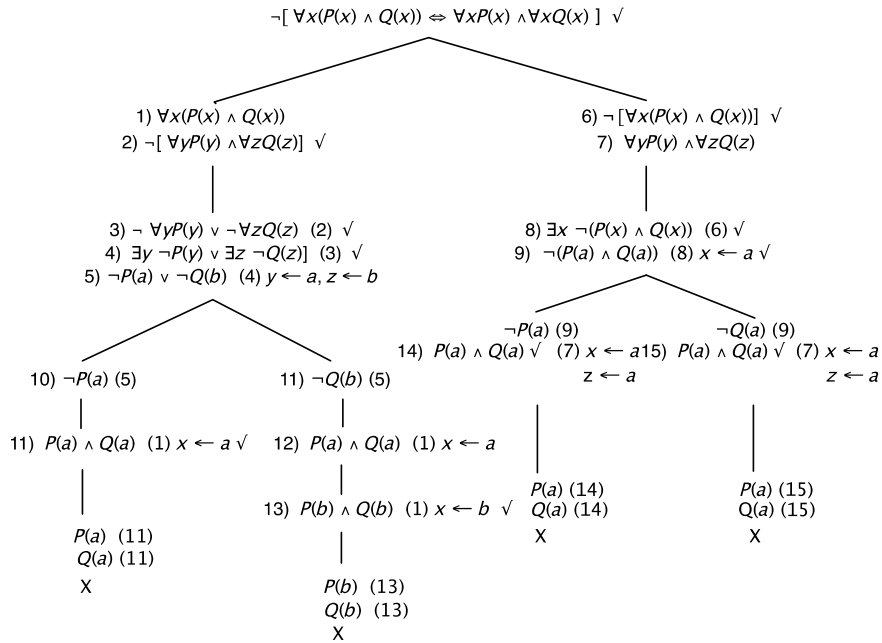
with:

$$l = (j - i) + n \times (i - 1) - (i \times (i - 1)) \div 2.$$

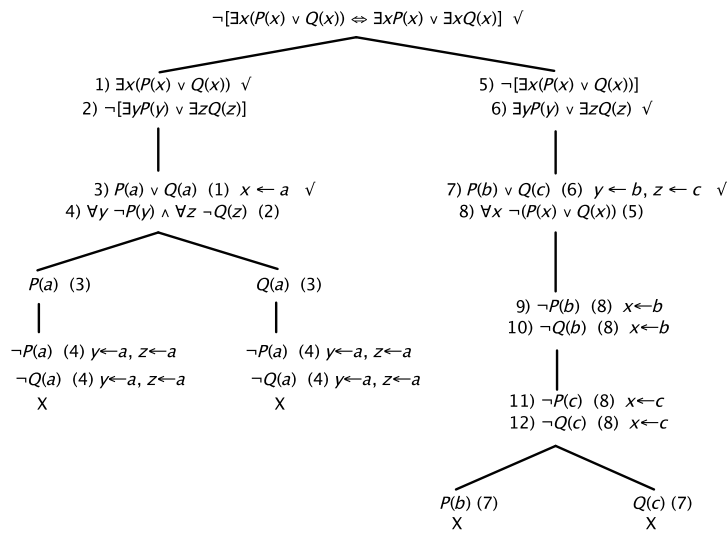
□

EXERCISE 5.3.–

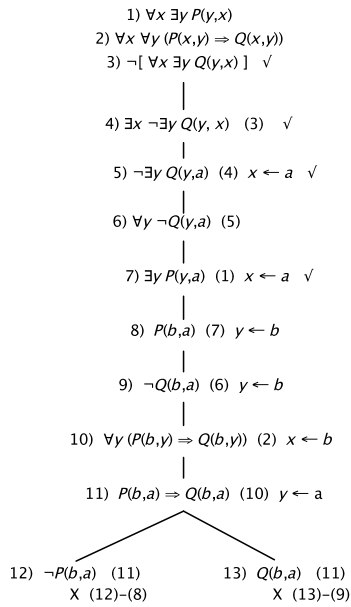
a)



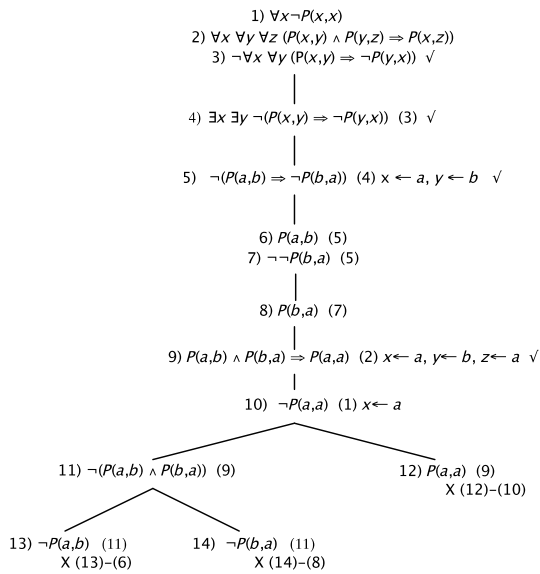
b)



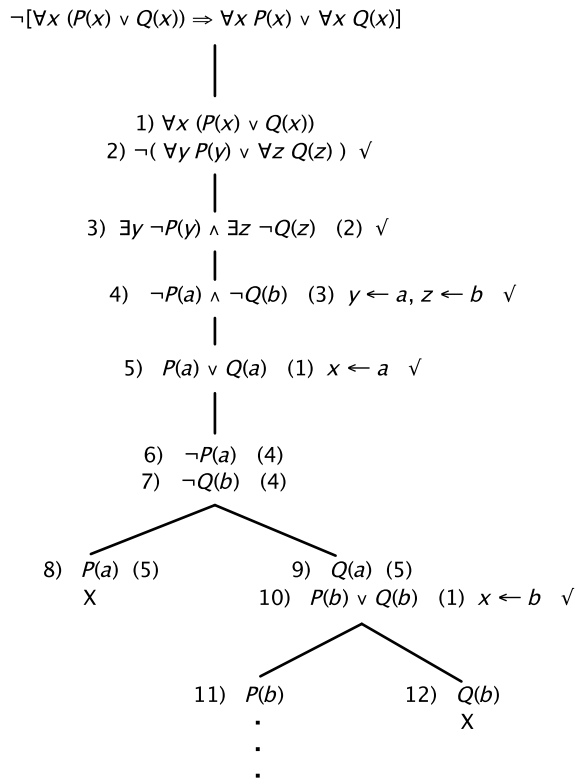
c)



d)



e)

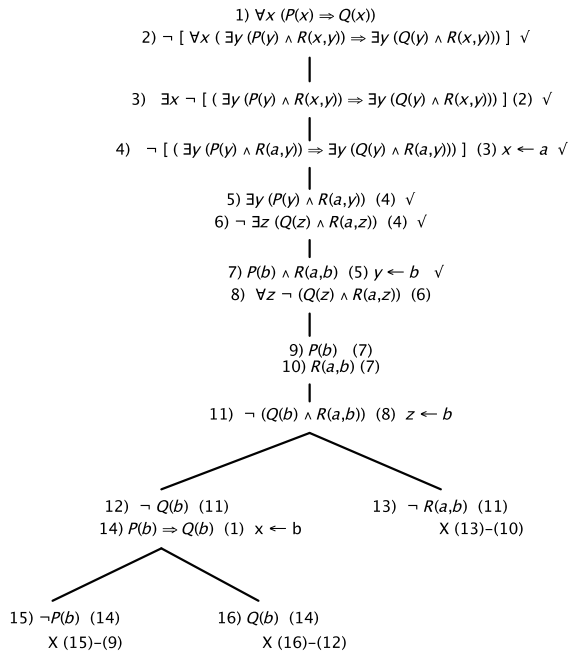


The method continues with  $P(c) \wedge Q(c)$ ,  $P(d) \wedge Q(d)$ , ... the tree will not be closed (this will not be detected by the method). □

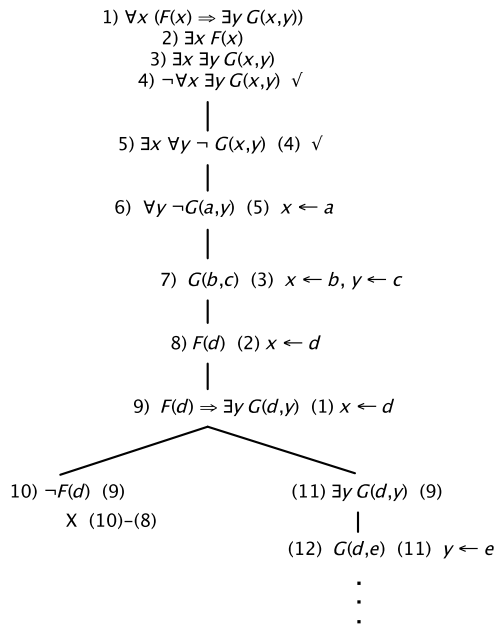
REMARK 12.13.– If we had not renamed the variables, i.e. if instead of (3) we had written  $\exists y \neg P(y) \wedge \exists y \neg Q(y)$  then we could have “proved the validity” of the given formula. □



f)



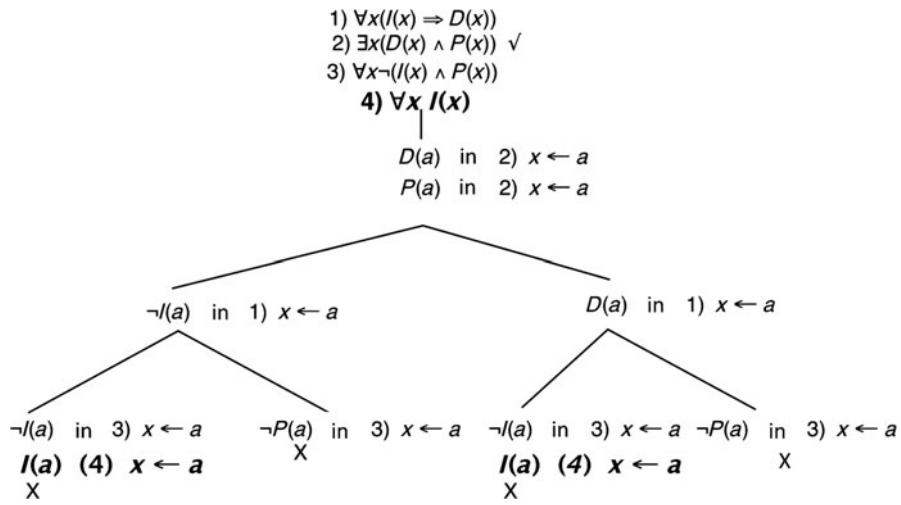
g)



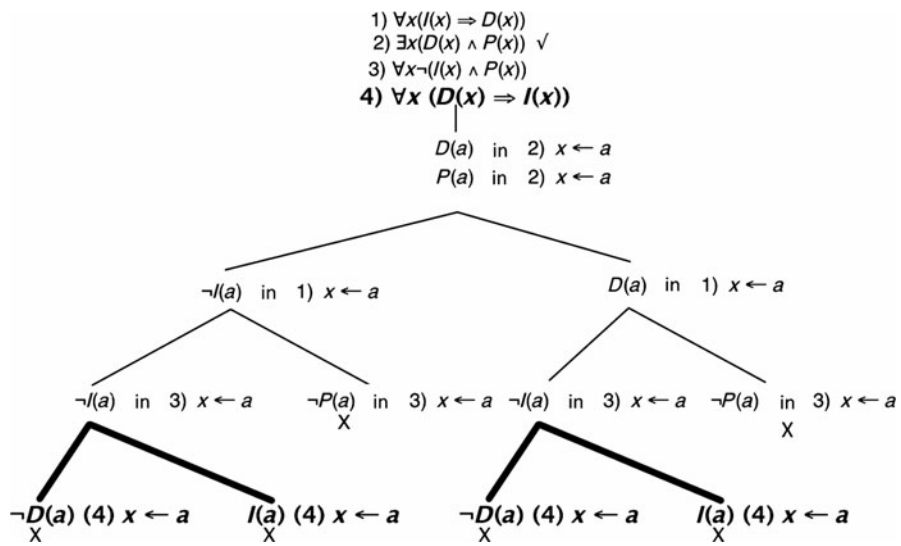


j) (See also section 8.3.)

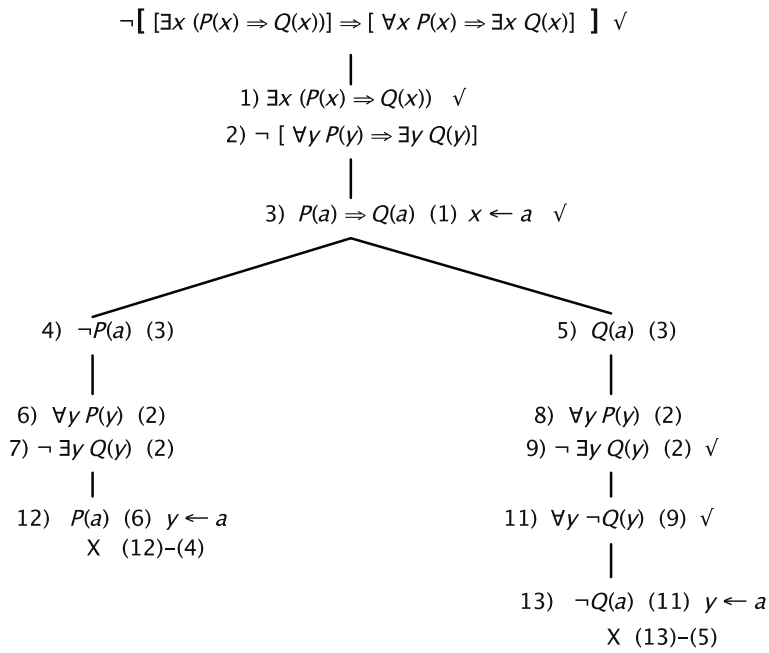
By examining the tree, we notice that we could close the tree by adding the formula  $\forall x I(x)$ .



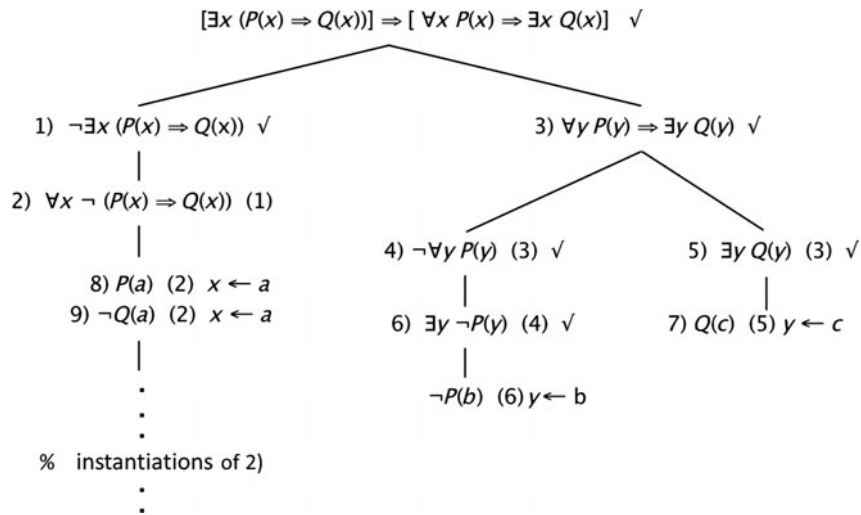
or  $\forall x(D(x) \Rightarrow I(x))$



k1) To verify the validity of the formula, we negate it.



k2) To try to construct a model of the formula, we do not negate it.



We can extract the following models from the *finite* open branches:

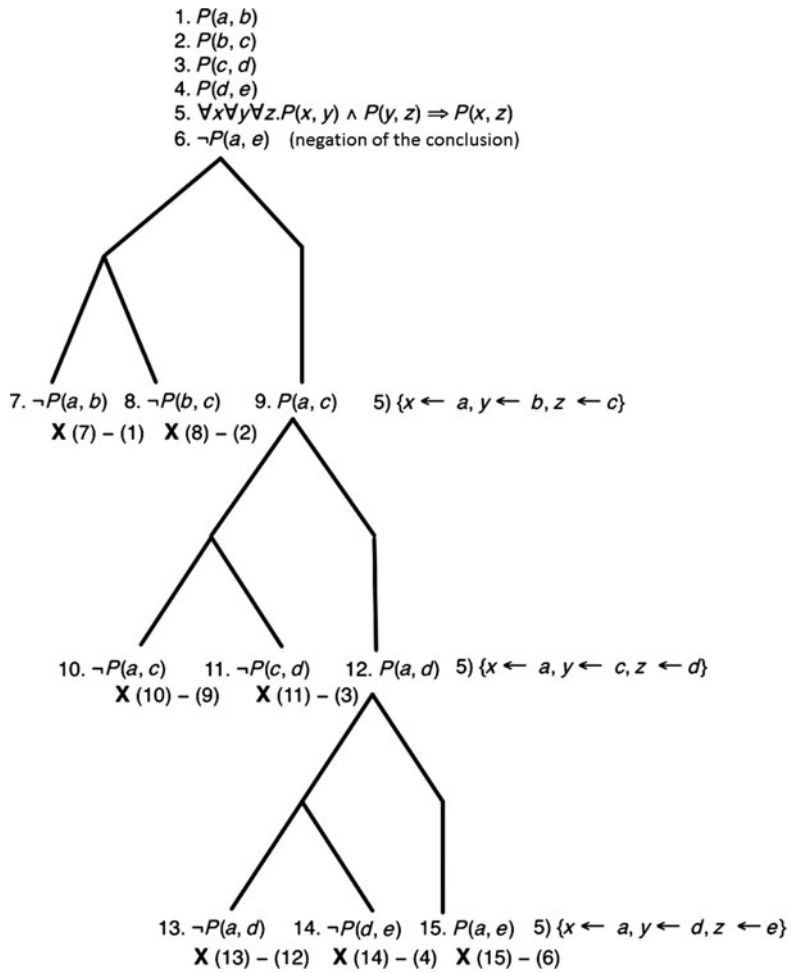
$$\{Q(c)\} \rightsquigarrow [ \quad ] \Rightarrow [ \quad \Rightarrow \mathbf{T} ]$$

$$\{\neg P(b)\} \rightsquigarrow [ \quad ] \Rightarrow [ \mathbf{F} \Rightarrow \quad ]$$

We can extract the following model from the *infinite* branch:

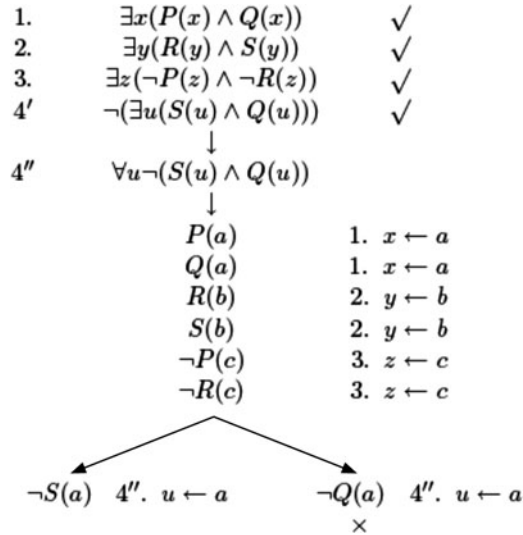
$$\{P(a), \neg Q(a)\} \rightsquigarrow [ \mathbf{T} \Rightarrow \mathbf{F} ] \Rightarrow [ \quad ]$$

1)



□

EXERCISE 5.4.– The tableau corresponding to the given reasoning:



It is clear that if:

– the cardinality of the domain of discourse is 1, i.e.  $a = b = c$ , then we obtain a closed tree ( $P(a), \neg P(c)$ );

– similarly if the cardinality of the domain of discourse is 2, i.e.  $a = b$  ( $S(b), \neg S(a)$ ) or  $a = c$  ( $P(a), \neg P(c)$ ) or  $b = c$  ( $R(b), \neg R(c)$ );

However, if the cardinality of the domain of discourse is 3, i.e.  $a \neq b$  and  $a \neq c$  and  $b \neq c$ , we obtain the following model from the set of formulas (1), (2), (3), (4)'':

$$\mathcal{M} = \{P(a), Q(a), R(b), S(b), \neg P(c), \neg R(c), \neg S(a)\}$$

( $P(b), Q(b), Q(c), R(a)$  can belong to  $\mathcal{M}$  or not)

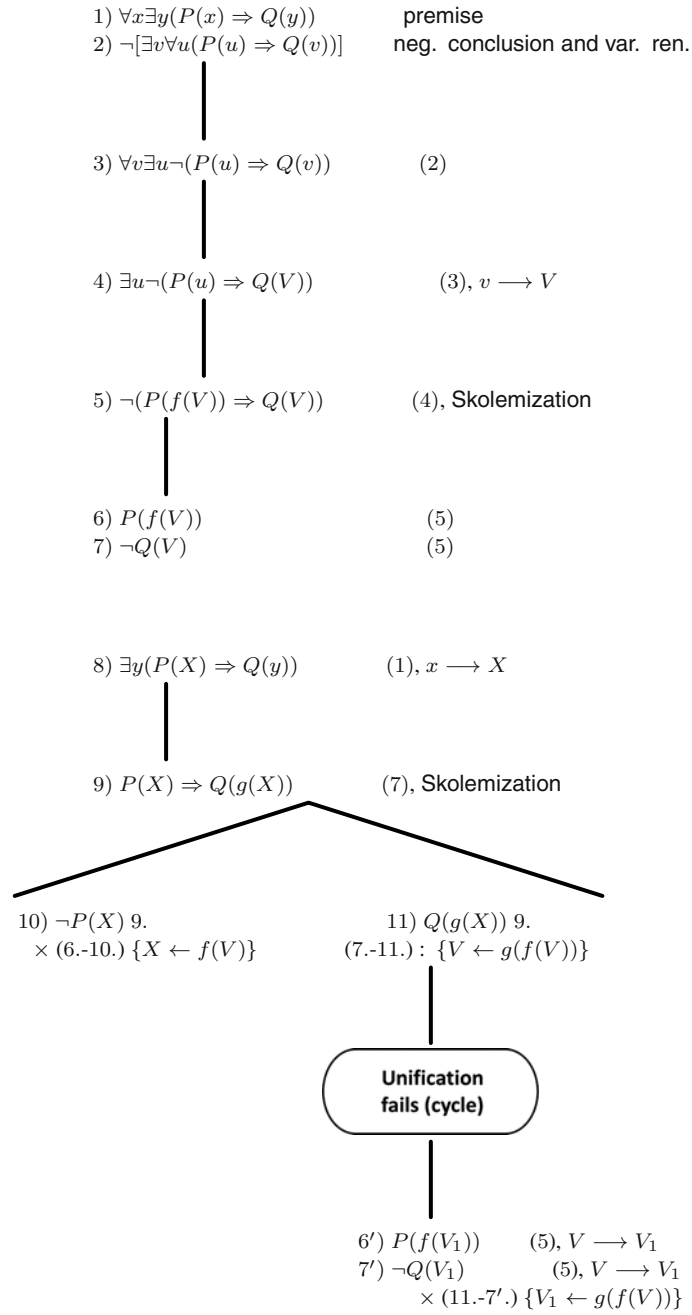
It is very simple to verify that this is a counter example of the reasoning on a domain of discourse of cardinality 3. By construction,  $\mathcal{M}$  is a model of the premises. About the conclusion:

- if  $x = a$ , then in  $\mathcal{M}$  we have  $S(a)$ : **F** and  $Q(a)$ : **T** ;

- if  $x = b$ , then we have  $\mathcal{M} S(b)$ : **T** and  $Q(b)$ : **F** ;

hence, in all cases, the conclusion is evaluated to **F**. □

EXERCISE 5.5.–



□

EXERCISE 5.6.–

$$\% P(f^0(a)) = P(a) ; Q(f^0(a)) = Q(a)$$

$$MH1 = \{Q(a), P(f^n(a))\} \quad n \in \mathbb{N} \% n = 0, 1, 2, \dots$$

i.e.:

$$P(a) \mapsto \mathbf{T}, Q(a) \mapsto \mathbf{T}, P(f(a)) \mapsto \mathbf{T}, Q(f(a)) \mapsto \mathbf{F}, P(f^2(a)) \mapsto \mathbf{T}, \\ Q(f^2(a)) \mapsto \mathbf{F}, P(f^3(a)) \mapsto \mathbf{T}, Q(f^3(a)) \mapsto \mathbf{F}, \dots$$

$$MH2 = \{P(f^n(a)), Q(f^n(a))\} \quad n = 2 \times i; \quad i \in \mathbb{N} \% n = 0, 2, 4, \dots$$

i.e.:

$$P(a) \mapsto \mathbf{T}, Q(a) \mapsto \mathbf{T}, P(f(a)) \mapsto \mathbf{F}, Q(f(a)) \mapsto \mathbf{F}, P(f^2(a)) \mapsto \mathbf{T}, \\ Q(f^2(a)) \mapsto \mathbf{T}, P(f^3(a)) \mapsto \mathbf{F}, Q(f^3(a)) \mapsto \mathbf{F}, \dots$$

$$MH3 = \{P(a), Q(a), P(f^n(a)), Q(f^n(a))\}; \quad n = 2 \times i + 1 \quad i \in \mathbb{N} \% \\ n = 1, 3, 5, \dots$$

i.e.:

$$P(a) \mapsto \mathbf{T}, Q(a) \mapsto \mathbf{T}, P(f(a)) \mapsto \mathbf{T}, Q(f(a)) \mapsto \mathbf{T}, P(f^2(a)) \mapsto \mathbf{F}, \\ Q(f^2(a)) \mapsto \mathbf{F}, P(f^3(a)) \mapsto \mathbf{T}, Q(f^3(a)) \mapsto \mathbf{T}, \dots$$

Note that  $M = MH1 \cap MH2 \cap MH3 = \{P(a), Q(a)\}$  is also a model of  $\mathcal{S}$ , as  $\mathcal{S}$  is a set of Horn clauses (see exercise 3.46).  $\square$

EXERCISE 5.7.– We are here in the case of example 5.16, wherein we had to apply the method of semantic tableaux to what we called “generic formulae”.

Without loss of generality and for the sake of clarity of the proof, we write the binary resolution rule under the form:

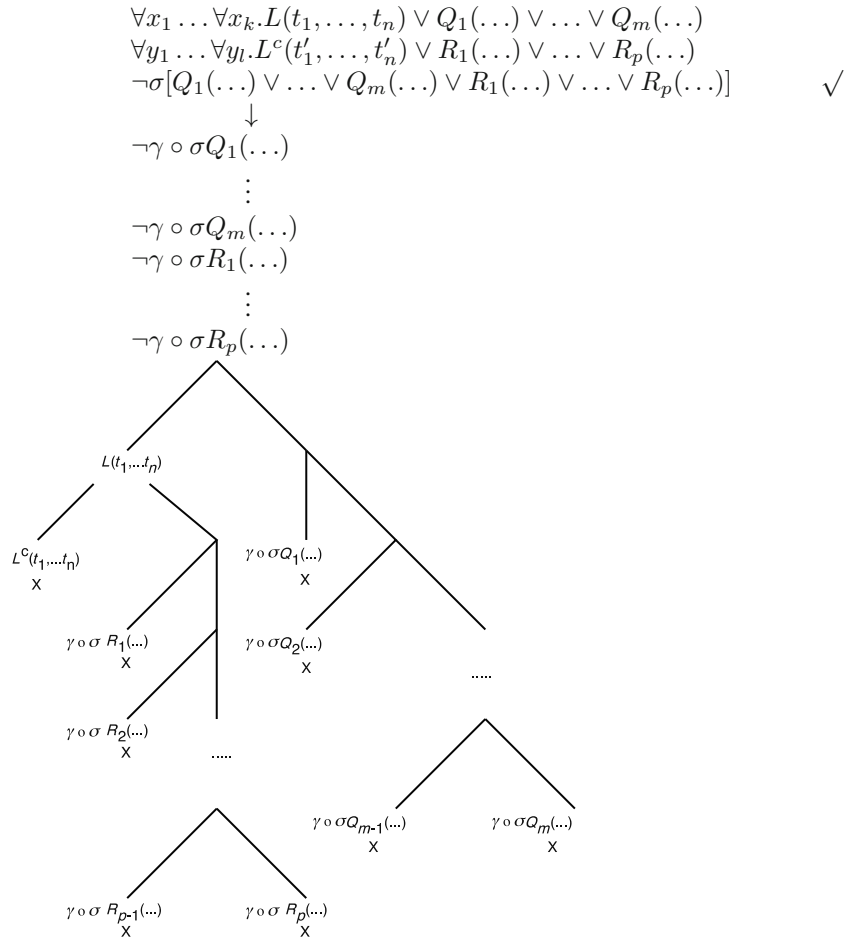
$$L(t_1, \dots, t_n) \vee Q_1(\dots) \vee \dots \vee Q_m(\dots) \quad L^c(t'_1, \dots, t'_n) \vee R_1(\dots) \vee \dots \vee R_p(\dots)$$

---


$$\sigma[Q_1(\dots) \vee \dots \vee Q_m(\dots) \vee R_1(\dots) \vee \dots \vee R_p(\dots)]$$

To prove that the resolvent is a logical consequence of the parent clauses, we construct the tableau below.





It should be clear that we transformed the disjunctions of literals into binary trees (which is required by our formalization of semantic tableaux) and that:

$$Var(L(t_1, \dots, t_n) \vee Q_1(\dots) \vee \dots \vee Q_m(\dots)) = \{x_1, \dots, x_k\}$$

$$Var(L^c(t'_1, \dots, t'_n) \vee R_1(\dots) \vee \dots \vee R_p(\dots)) = \{y_1, \dots, y_l\}$$

The use of substitution  $\gamma \circ \sigma$  (instead of simply  $\sigma$ ) can be explained by the fact that possibly,  $x_i \leftarrow y_j \in \sigma$  for  $i \in [1, k]$ ,  $j \in [1, l]$  (or  $y_j \leftarrow x_i$ ). In this case  $x_i \leftarrow y_j \in \gamma$ .

Similarly for the  $x_i, y_j \notin dom(\sigma)$  ( $x_i, y_j$  are variables of the parent clauses)  $x_i \leftarrow a \in \gamma, y_j \leftarrow a \in \gamma$  (instead of  $a$ , any other closed term could be used).  $\gamma$ , which

can also take into account the possible renamings of the variables of the resolvent, is thus used exclusively to close branches because of contradictions between *closed literals* (i.e. containing only closed terms), thus ensuring that the rules of the method of semantic tableaux are applied to formulas that are propositional (up to the syntax).

$\gamma \circ \sigma R_i(\dots)$  ( $1 \leq i \leq p$ ) and:

$\gamma \circ \sigma Q_j(\dots)$  ( $1 \leq j \leq m$ )

These are constant instances corresponding to the literals of the parent clauses whose variables are quantified universally (and as usual we chose the instances to be able to close the branches).  $\square$

EXERCISE 5.8.– First we must transform the premises and the negation of the conclusion into clausal form (the labels on  $\longrightarrow$  refer to the rules used in the proof of theorem 5.3).

$$\begin{aligned} \exists x[P(x) \wedge \forall y(R(y) \Rightarrow S(x, y))] &\xrightarrow{3'} \exists x[\forall y(P(x) \wedge (R(y) \Rightarrow S(x, y)))] \xrightarrow{Skol.} \\ \forall y(P(a) \wedge (R(y) \Rightarrow S(a, y))) &\xrightarrow{3'} P(a) \wedge \forall y(R(y) \Rightarrow S(a, y)) \longrightarrow P(a) \wedge \\ \forall y(\neg R(y) \vee S(a, y)) &\quad \% \text{ i.e. two clauses} \end{aligned}$$

$$\begin{aligned} \forall x[P(x) \Rightarrow \forall y(Q(y) \Rightarrow \neg S(x, y))] &\xrightarrow{5'} \forall x[\forall y(P(x) \Rightarrow (Q(y) \Rightarrow \neg S(x, y)))] \\ \longrightarrow \forall x \forall y(\neg P(x) \vee \neg Q(y) \vee \neg S(x, y)) &\quad \% \text{ one clause.} \end{aligned}$$

We rename the variables to avoid any confusion:

$$\neg P(z) \vee \neg Q(u) \vee \neg S(z, u)$$

Negation of the conclusion:

$$\begin{aligned} \neg[\forall x(R(x) \Rightarrow \neg Q(x))] &\longrightarrow \exists x \neg[R(x) \Rightarrow \neg Q(x)] \xrightarrow{Skol.} \neg(R(b) \Rightarrow \neg Q(b)) \longrightarrow \\ R(b) \wedge Q(b) & \end{aligned}$$

$\% \text{ i.e. two clauses}$

To prove that the reasoning is correct, we must refute clauses 1. to 5. below:

- 1)  $P(a)$
- 2)  $\neg R(y) \vee S(a, y)$
- 3)  $\neg P(z) \vee \neg Q(u) \vee \neg S(z, u)$
- 4)  $R(b)$
- 5)  $Q(b)$

- 6)  $S(a, b)$  (2, 1) – (4, 1)  $\{y \leftarrow b\}$   
 7)  $\neg P(a) \vee \neg Q(b)$  (3, 3) – (6, 1)  $\{z \leftarrow a, u \leftarrow b\}$   
 8)  $\neg Q(b)$  (1, 1) – (7, 1)  
 9)  $\square$  (5, 1) – (8, 1)

□

## EXERCISE 5.9.–

- 1)  $\neg P(x, y, z, s) \vee P(z, y, z, f(x, z, s))$   
 2)  $\neg P(x, y, x, s) \vee P(y, y, y, g(x, y, s))$   
 3)  $\neg P(b, b, b, s) \vee R(h(s))$   
 4)  $P(a, b, c, d)$   
 5)  $\neg R(u)$   
 6)  $\neg P(b, b, b, s_3)$  (5, 1) – (3, 2)  $\{u \leftarrow h(s_3)\}$   
 7)  $\neg P(x_2, b, x_2, s_2)$  (6, 1) – (2, 2)  $\{s_3 \leftarrow g(x_2, b, s_2), y \leftarrow b\}$   
 8)  $\neg P(x_1, b, z_1, s_1)$  (7, 1) – (1, 2)  $\{s_2 \leftarrow f(x_1, z_1, s_1), \dots\}$   
 9)  $\square$  (8, 1) – (4, 1)  $\{x_1 \leftarrow a, z_1 \leftarrow c, s_1 \leftarrow d\}$

by replacing:

$$s_2 \leftarrow f(a, c, d)$$

$$s_3 \leftarrow g(c, b, f(a, c, d))$$

$$u \leftarrow h(g(c, b, f(a, c, d)))$$

if we had asked the question

$$\neg R(u) \vee \text{Answer}(u)$$

instead of  $\square$  we would have generated (see exercise 3.34):

$$\text{Answer}(h(g(c, b, f(a, c, d))))$$

which translates, in accordance with the interpretation that we gave to the function symbols (in particular to the constants) into: *the monkey climbs on the chair after having walked from a to c starting from d, and carried the chair from c to b.* □

## EXERCISE 5.10.–

a)

- |   |   |
|---|---|
| 1) $\neg P(x) \vee Q(x) \vee R(x, f(x))$    |   |
| 2) $\neg P(u) \vee Q(u) \vee S(f(u))$       |   |
| 3) $T(a)$                                   |   |
| 4) $P(a)$                                   |   |
| 5) $\neg R(a, y) \vee T(y)$                 |   |
| 6) $\neg T(z) \vee \neg Q(z)$               |   |
| 7) $\neg T(w) \vee \neg S(w)$               |   |
| 8) $\neg Q(a)$                              | (3, 1) – (6, 1) $\{z \leftarrow a\}$                    |
| 9) $\neg P(a) \vee S(f(a))$                 | (8, 1) – (2, 2) $\{x \leftarrow a\}$                    |
| 10) $\neg P(a) \vee Q(a) \vee T(f(a))$      | (1, 3) – (5, 1) $\{x \leftarrow a, y \leftarrow f(a)\}$ |
| 11) $\neg P(a) \vee Q(a) \vee \neg S(f(a))$ | (10, 3) – (7, 1) $\{w \leftarrow f(a)\}$                |
| 12) $Q(a) \vee S(f(a))$                     | (2, 1) – (4, 1) $\{u \leftarrow a\}$                    |
| 13) $S(f(a))$                               | (12, 1) – (8, 1)  |
| 14) $\neg P(a) \vee Q(a)$                   | (13, 1) – (11, 3)                                       |
| 15) $Q(a)$                                  | (14, 1) – (4, 1)  |
| 16) $\square$                               | (15, 1) – (8, 1)  |

b)

- |  |  |
|--|--|
| 1) $P(x) \vee P(a) \vee R(x)$  |  |
| 2) $\neg P(y) \vee Q(y)$   |  |
| 3) $\neg R(z) \vee Q(z) \vee M(z)$                                     |  |
| 4) $\neg Q(w) \vee \neg R(w)$  |  |
| 5) $\neg M(v) \vee \neg R(v) \vee \neg R(a)$                           |  |
| 6) $P(a) \vee R(y_6) \vee Q(y_6)$                                      | (1, 1) – (2, 1) $\{x \leftarrow y_2\}$ |
| 7) $P(z_7) \vee P(a) \vee Q(z_7) \vee M(z_7)$                          | (1, 3) – (3, 3) $\{x \leftarrow z_3\}$ |
| 8) $P(v_8) \vee P(a) \vee \neg M(v_8) \vee \neg R(v_8) \vee \neg R(a)$ | (1, 3) – (5, 2) $\{x \leftarrow v_5\}$ |
| 9) $P(a) \vee P(a) \vee \neg M(v_9) \vee \neg R(v_9)$                  | (1, 3) – (5, 3) $\{x \leftarrow a\}$   |
| 10) $\neg P(w_{10}) \vee \neg R(w_{10})$                               | (2, 2) – (4, 1) $\{y \leftarrow w_4\}$ |
| 11) $\neg R(w_{11}) \vee M(z_{11}) \vee \neg R(w_{11})$                | (3, 2) – (4, 1) $\{z \leftarrow w_4\}$ |
| 12) $\neg R(v_{12}) \vee Q(v_{12}) \vee \neg R(v_{12}) \vee R(a)$      | (3, 3) – (5, 1) $\{z \leftarrow v_5\}$ |

⋮

□

REMARK 12.16.– We systematically rename the variables of the resolvents.

These are sets of clauses that belong to a decidable fragment of FOL (the clauses do not contain any functional symbol). After a while, no new clause is generated (up to a renaming of the variables) and since  $\square$  is not generated, we conclude that the set of clauses (1) to (5) is satisfiable.  $\square$

EXERCISE 5.11.–

To apply the resolution rule, we must transform into clausal form:

from  $\forall x \exists y P(x, y)$ , after Skolemization we obtain  $\forall x P(x, f(x))$ .

For the conclusion,  $\exists z \forall u P(z, u)$ :

negation  $\rightarrow$  Skolemization:  $\neg[\exists z \forall u P(z, u)] \rightarrow \forall z \exists u \neg P(z, u) \rightarrow \forall z \neg P(z, g(z))$

The set of clauses to consider:  $\{P(x, f(x)), \neg P(z, g(z))\}$

The resolution rule cannot be applied (the unification algorithm fails with *clash*),  $\square$  cannot be obtained: the answer is therefore no.  $\square$

EXERCISE 5.12.–

i) This exercise shows how to encode the axioms and the inference rule of  $S_1$  in clausal form and prove theorems using the resolution rule.

We write (A1), (A2), and (A3) in clausal form. The connectives  $\Rightarrow$  and  $\neg$  are written:

$i(x, y): x \Rightarrow y$

$n(x): \neg x$

1)  $P(i(x, i(y, x)))$

2)  $P(i(i(x, i(y, z)), i(i(x, y), i(x, z))))$

3)  $P(i(i(n(y), n(x)), i(i(n(y), x), y)))$

*The axioms are provable.*

To translate MP, we say: *If  $x$  is provable and  $x \Rightarrow y$  (i.e.  $i(x, y)$ ) is provable, then  $y$  is provable:*

4)  $\neg P(i(x, y)) \vee \neg P(x) \vee P(y)$

We ask the question  $\vdash^? A \Rightarrow A$  (see example 3.9). To better recall that  $A$  is a (meta) variable that denotes an arbitrary wff, we write  $\vdash^? v \Rightarrow v$  ( $v$  denotes a variable).

To show that  $v \Rightarrow v$  is provable in  $S_1$  (encoded  $P(i(v, v))$ ), we must generate  $\square$ , applying the resolution rule on 1–4 and:

$$5) \neg P(i(v, v))$$

As usual, we identify the variables in each clause with the corresponding index and rename the variables of the resolvents. We use a *linear strategy*.

$$6) \neg P(i(x_6, i(v_6, v_6))) \vee \neg P(x_6) \quad (5, 1) - (4, 3)$$

$$\{y_4 \leftarrow i(v_5, v_5)\}$$

$$7) \neg P(i(i(x_7, i(y_7, x_7)), i(v_7, v_7))) \quad (6, 2) - (1, 1)$$

$$\{x_6 \leftarrow i(x_1, i(y_1, x_1))\}$$

$$8) \square \quad (7, 1) - (2, 1)$$

$$\{x_7 \leftarrow z_2, y_7 \leftarrow z_2, x_2 \leftarrow z_2, y_2 \leftarrow z_2, v_7 \leftarrow i(z_2, z_2)\}$$

ii)

We proceed by *reductio ad absurdum*. We assume that

$A \Rightarrow A$  is not provable (5)

This implies that  $A$  is not provable, or that  $A \Rightarrow (B \Rightarrow B)$  is not provable (6);

and that, in turn,  $(A \Rightarrow (B \Rightarrow A)) \Rightarrow (C \Rightarrow C)$  is not provable (7)

As  $A, B, C$  are meta-variables (i.e. variables that can be replaced by arbitrary wffs), to avoid any confusion, we shall name them  $X, Y, Z$  respectively, so that

$(X \Rightarrow (Y \Rightarrow X)) \Rightarrow (Z \Rightarrow Z)$  is not provable.

One instance is (with  $X \leftarrow C, Y \leftarrow C, Z \leftarrow C \Rightarrow C$ ):

$$(*) C \Rightarrow (C \Rightarrow C) \Rightarrow ((C \Rightarrow C) \Rightarrow (C \Rightarrow C))$$

(\*) is also an instance of (A2)

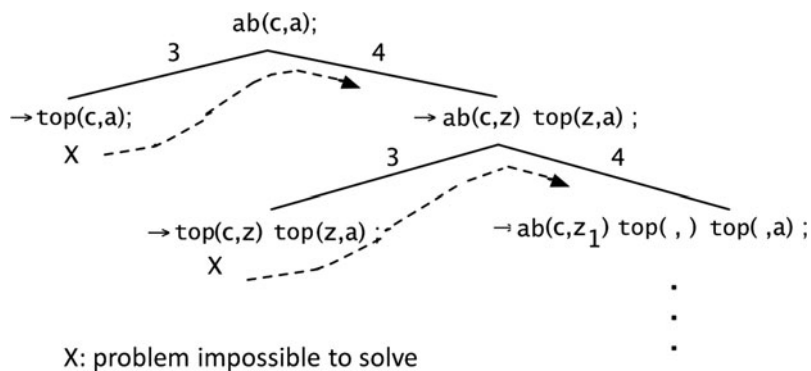
But this is a contradiction, as all the instances of an axiom are (by definition of a proof) provable.  $\square$

EXERCISE 6.1.–

a)

- 1)  $\text{top}(a, b) \rightarrow;$
- 2)  $\text{top}(b, c) \rightarrow;$
- 3)  $\text{ab}(x, y) \rightarrow \text{top}(x, y);$
- 4)  $\text{ab}(u, v) \rightarrow \text{ab}(u, z) \text{ top}(z, v);$

$\text{ab}(c, a);$  % the question

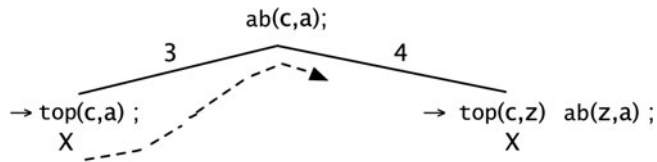


The execution does not halt.

If we consider the clause (\*\*) instead of (\*) :

- 1)  $\text{top}(a, b) \rightarrow;$
- 2)  $\text{top}(b, c) \rightarrow;$
- 3)  $\text{ab}(x, y) \rightarrow \text{top}(x, y);$
- 4)  $\text{ab}(u, v) \rightarrow \text{top}(u, z) \text{ ab}(z, v);$

$ab(c, a); \% \text{ the question}$



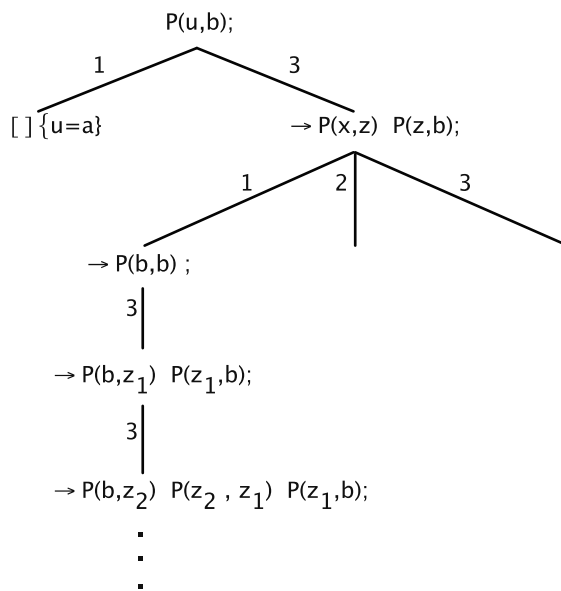
**X: problem impossible to solve**

The program halts and answers No.

b)

- 1)  $P(a, b) \rightarrow;$
- 2)  $P(c, a) \rightarrow;$
- 3)  $P(x, y) \rightarrow P(x, z) P(z, y);$

$P(u, b); \% \text{ the question}$



Of course this problem has two solutions  $u = a$  and  $u = c, \dots$  but only the first one is found. To try finding all the solutions, we can swap the orders of clauses 1 and 2, or



swap literals  $P(x, z)$  and  $P(z, y)$  in clause 3. In both cases, after having found both solutions, the program does not halt.

It is suggested to use the following technique:

- 1)  $P(a, b) \rightarrow;$
- 2)  $P(c, a) \rightarrow;$
- 3)  $Q(x, y) \rightarrow P(x, y);$
- 4)  $Q(x, y) \rightarrow P(x, z) \quad Q(z, y);$

$Q(u, b)$  % the question □

DIGRESSION 12.1.– When transforming programs, it is natural to wonder whether they are equivalent, and in the case of logic programming, this boils down to ask whether one program is a logical consequence of another one.

We seize this opportunity to show the usefulness of automated theorem provers (or proof assistants) for programming. We have used one of the best-known provers based on the resolution method (*Prover9*), which contains a tool capable of constructing finite models (*Mace4*).

If we let P1 stand for the initial program and P2 stand for the suggested program, we show that P1 **is not** a logical consequence of P2 (by exhibiting a counter-model constructed by *Mace4*). However, if the implication of clause 3. of the modified program is replaced by an equivalence, which yields, say, P2', then we give a proof that P1 **is** a logical consequence of P2'.

```

===== INPUT =====
assign(report_stderr,2).
set(ignore_option_dependencies).
if(Prover9).
  \%% Conditional input included.
  assign(max_seconds,6C).
end_if.
if(Mace4).
  \%% Conditional input omitted.
end_if.

formulas(assumptions).
P(a,b) & P(c,a) & (all x all y (P(x,y) -> Q(x,y))) & (all x all y all z (P(x,z) & Q(z,y) -> Q(x,y))).
end_of_list.

formulas(goals).
P(a,b) & P(c,a) & (all x all y all z (P(x,z) & P(z,y) -> P(x,y))).
end_of_list.

===== end of input =====

```

```

\% Enabling option dependencies (ignore applies only on input).

===== PROCESS NON-CLAUSAL FORMULAS =====

\% Formulas that are not ordinary clauses:
1 P(a,b) & P(c,a) & (all x all y (P(x,y) -> Q(x,y))) & (all x all y all z (P(x,z) & Q(z,y) -> Q(x,y)))
# label(non_clause). [assumption].
2 P(a,b) & P(c,a) & (all x all y all z (P(x,z) & P(z,y) -> P(x,y)))
# label(non_clause) # label(goal). [goal].

===== end of process non-clausal formulas =====

===== PROCESS INITIAL CLAUSES =====

\% Clauses before input processing:

formulas(usable).
end_of_list.

formulas(sos).
P(a,b). [clausify(1)].
P(c,a). [clausify(1)].
-P(x,y) | Q(x,y). [clausify(1)].
-P(x,y) | -Q(y,z) | Q(x,z). [clausify(1)].
-P(a,b) | -P(c,a) | P(c1,c3). [deny(2)].
-P(a,b) | -P(c,a) | P(c3,c2). [deny(2)].
-P(a,b) | -P(c,a) | -P(c1,c2). [deny(2)].
end_of_list.

formulas(denodulators).
end_of_list.

Counterexample:

interpretation( 2, [number = 1,seconds = C], [
  function(a, [C]),
  function(b, [C]),
  function(c, [C]),
  function(c1, [1]),
  function(c2, [1]),
  function(c3, [C]),
  relation(P(_,_), [
    1,1,
    1,C]),
  relation(Q(_,_), [
    1,1,
    1,1])]).

===== PROOF =====

\% ----- Comments from original proof -----
\% Proof 1 at C.CC (+ C.CC) seconds.
\% Length of proof is 1E.
\% Level of proof is 4.
\% Maximum clause weight is 9.
\% Given clauses 1C.

1 P(a,b) & P(c,a) & (all x all y (P(x,y) <-> Q(x,y))) & (all x all y all z (P(x,z) & Q(z,y) -> Q(x,y)))

```

```

# label(non_clause). [assumption].
2 P(a,b) & P(c,a) & (all x all y all z (P(x,z) & P(z,y) -> P(x,y)))
# label(non_clause) # label(goal). [goal].
3 P(a,b). [clausify(1)].
4 P(c,a). [clausify(1)].
5 -P(x,y) | Q(x,y). [clausify(1)].
6 P(x,y) | -Q(x,y). [clausify(1)].
7 -P(x,y) | -Q(y,z) | Q(x,z). [clausify(1)].
8 -P(a,b) | -P(c,a) | P(c1,c3). [deny(2)].
9 P(c1,c3). [copy(8),unit_del(a,3),unit_del(b,4)].
10 -P(a,b) | -P(c,a) | P(c3,c2). [deny(2)].
11 P(c3,c2). [copy(10),unit_del(a,3),unit_del(b,4)].
12 -P(a,b) | -P(c,a) | -P(c1,c2). [deny(2)].
13 -P(c1,c2). [copy(12),unit_del(a,3),unit_del(b,4)].
17 Q(c3,c2). [ur(5,a,11,a)].
18 -Q(c1,c2). [resolve(13,a,6,a)].
20 $F. [ur(7,a,9,a,c,18,a),unit_del(a,17)].

===== end of proof =====

```

□

EXERCISE 6.2.– The intended meaning of the terms:

a: 0

s(x): successor of  $x$  (in  $\mathbb{N}$ )

a)

add(a, x, x)  $\rightarrow$ ;

add(s(x), y, s(z))  $\rightarrow$  add(x, y, z);

b)

mult(a, x, a)  $\rightarrow$ ;

mult(s(x), y, u)  $\rightarrow$  mult(x, y, z) add(z, y, u);

c)

less(a, s(x))  $\rightarrow$ ;

less(s(x), s(y))  $\rightarrow$  less(x, y);

d)

divides(x, y)  $\rightarrow$  not(eq(x, 0)) division(y, x, z);

division(y, x, z)  $\rightarrow$  mult(z, x, y);

% not: see section 6.3.2

e)

prime(x) → not(eq(x,0)) not(eq(y,1)) less(y,x) not(divides(y,x))

% eq: see exercise 6.10

% not(eq(y,0)) not necessary because in divides □

EXERCISE 6.3.– The intended meanings of the terms:

a: 0

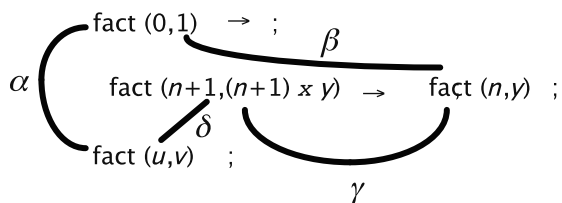
s(x): successor of x (in  $\mathbb{N}$ )

fib(a, a) →;

fib(s(a), s(a)) →;

fib(s(s(x)), u) → fib(x, y) fib(s(x), z) add(y, z, u); □

EXERCISE 6.4.–



The regular expression that characterizes the sequence of applications of the resolution rule with the strategy used by the interpreter PL (each application is represented by the used mgu):

$\alpha\delta\gamma^*\beta$

The application of  $\alpha$  gives the first solution, the rest of the regular expression gives the other solutions. □

EXERCISE 6.5.–

a)

append([ ], x, x) →;

`append([u | x], y, [u | z]) → append(x, y, z);`

b)

`reverse([ ], [ ]) →;`

`reverse([t | x], z) → reverse(x, y) append(y, [t], z);`

c)

`palindrome(x) → reverse(x, x);`

d)

`member(x, [x | y]) →;`

`member(x, [z | y]) → member(x, y);`

Other definition (program):

`member(x, y) → append(u, [x | z], y);`

e)

`subset([ ], y) →;`

`subset([z | x], y) → member(z, y) subset(x, y);`

f)

We give three definitions (programs).

i)

`consec(x, y, [x, y | u]) →;`

`consec(x, y, [v | u]) → consec(x, y, u);`

ii)

`position(x, n, y)`: element `x` is at position `n` in list `y`

`position(u, 1, [u | x]) →;`

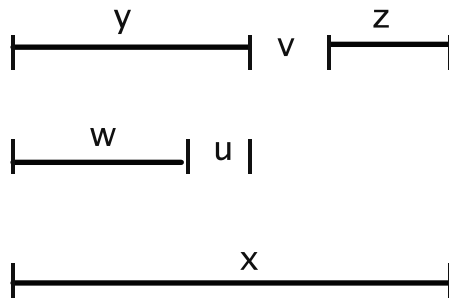
`position(u, i + 1, [v | x]) → position(u, i, x);`

$\text{consec}(u, v, x) \rightarrow \text{position}(u, i, x) \text{ position}(v, i + 1, x);$

iii)

$\text{consec}(u, v, x) \rightarrow \text{append}(y, [v | z], x) \text{ append}(w, [u], y);$

Idea:



□

EXERCISE 6.6.– We first define the predicate partition

$\text{partition}(\text{list\_}L, \text{element\_of\_}L(x), \text{list\_of\_smaller\_elements\_than\_}x\_in\_L, \text{list\_of\_greater\_elements\_than\_}x\_in\_L)$

$\text{partition}([], x, [], []) \rightarrow ;$

$\text{partition}([x|X], y, [x|P], G) \rightarrow x \leq y \text{ partition}(X, y, P, G) ;$

$\text{partition}([x|X], y, P, [x|G]) \rightarrow x > y \text{ partition}(X, y, P, G) ;$

$\text{quicksort}([], []) \rightarrow ;$

$\text{quicksort}([x|X], Y) \rightarrow \text{partition}(X, x, LT, GT)$

$\text{quicksort}(LT, P) \text{ quicksort}(GT, G) \text{ append}(P, [x|G], Y);$

$\text{append}([], X, X) \rightarrow ;$

$\text{append}([U|X], Y, [U|Z]) \rightarrow \text{append}(X, Y, Z) ;$

□

EXERCISE 6.7.–

```

bubblesort(X,X) → ;
bubblesort(X,Y) → append (U, [a,b|Z], X)
% We identify adjacent elements:
      append (U, [b,a|Z], T)
      bubblesort(T,Y) {a => b};

% {a => b} is a constraint whose meaning is obvious. It could be replaced by a
predicate similar to that of exercise 6.2 (c).

append([ ],X,X) → ;
append([U|X], Y, [U|Z ]) → append(X, Y, Z) ;

```

□

EXERCISE 6.9.–

a)

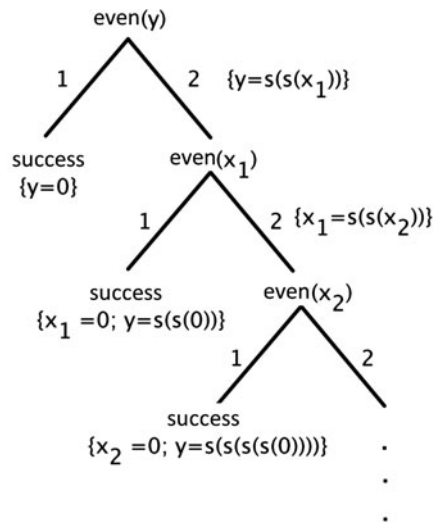
woman(adam)   2 not(man(adam))   1 not(success)   failure	woman(eve)   2 not(man(eve))   1 not(failure)   success
---	---

```

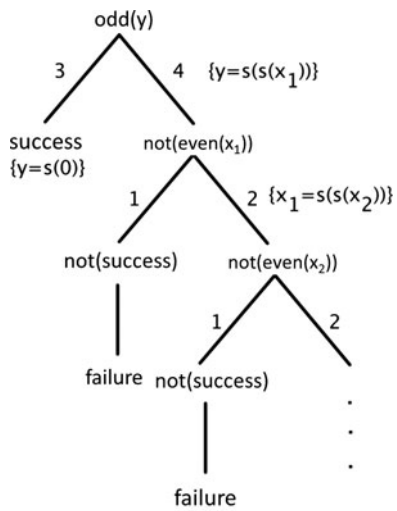
      woman(x) eq(x,eve)
      | 2
      not(man(u)) eq(u,eve)
      | 1
      not(man(adam)) eq(adam,eve)
      | 1
      not(success) eq(adam,eve)
      | 1
      failure

```

b)



We obtain:  $\{y=0, y=s(s(0)), y=s(s(s(0))), \dots\}$



We obtain:  $\{y=s(0)\}$

□



EXERCISE 6.11.–

$$\text{merge}(X, [], X) \rightarrow / ;$$

$$\text{merge}([], X, X) \rightarrow / ;$$

$$\text{merge}([x|X, [y|Y], [x|Z]) \rightarrow x < y / \text{merge}(X, [y|Y], Z) ;$$

$$\text{merge}([x|X, [y|Y], [x|y|Z]) \rightarrow x = y / \text{merge}(X, Y, Z) ;$$

$$\text{merge}([x|X, [y|Y], [y|Z]) \rightarrow x > y / \text{merge}([x|X], Y, Z) ; \quad \square$$

EXERCISE 8.1.– We prove properties that will permit us to simplify the answers to points (a) and (b).

We choose to represent clauses as sets of literals and recall that the sets of variables of distinct clauses are disjoint (see definition 5.10).

Property 1: let  $\sigma$  denote a substitution:

$$\text{if } C \leq_s D \text{ then } C \leq_s \sigma D \quad \% \text{ for an arbitrary } \sigma$$

PROOF.– By definition of a subsumption, there exists a substitution  $\theta$  such that:

$$\theta C \subseteq D.$$

The application of a substitution to a clause cannot change the number of its literals (see definition 5.12); hence, by applying an arbitrary substitution  $\sigma$  to  $\theta C$  and to  $D$ , the set inclusion cannot change:  $\sigma$  will act in the same way on  $\theta C$  and on the corresponding subset of literals in  $D$  (as it is the same!). We can thus write:

$$\sigma \theta C \subseteq \sigma D$$

there therefore exists a substitution (i.e.  $\sigma \circ \theta$ ) such that:

$$(\sigma \circ \theta)C \subseteq \sigma D$$

hence (definition of a subsumption):

$$C \leq_s \sigma D \quad (\text{for an arbitrary } \sigma).$$

We have the following immediate consequence.

Property 2: let  $C$  and  $D$  denote two clauses such that  $Var(D) = \{y_1, y_2, \dots, y_n\}$  and let  $\gamma$  be the substitution:

$$\gamma = \{y_1 \leftarrow a_1, y_2 \leftarrow a_2, \dots, y_n \leftarrow a_n\}$$

where the  $a_i$ 's ( $1 \leq i \leq n$ ) are constants *distinct* from those in  $C$  and  $D$  (intuition: the constants occurring in  $C$  and  $D$  already have an *intended meaning*, and to reuse them could change this meaning),

$$\text{if } C \leq_s \sigma D, \text{ then } C \leq_s \gamma D$$

PROOF.– Trivial, by applying property 1 with  $\gamma$  instead of  $\sigma$ .

The following property is a property that will be very useful to answer points (a) and (b).

Property 3: let  $\gamma$  be the substitution of property 2.

$$C \leq_s D \text{ iff } C \leq_s \gamma D$$

PROOF.– only if property 2

if

$$C \leq_s \gamma D$$

there therefore exists a substitution  $\theta$  such that:

$$\theta C \subseteq \gamma D$$

$$\text{Assume } Var(C) = \{x_1, x_2, \dots, x_m\}$$

and  $\theta = \{x_1 \leftarrow t_1, x_2 \leftarrow t_2, \dots, x_m \leftarrow t_m\}$  ( $x_i \in dom(\theta)$  and  $t_i$  is a closed term for  $1 \leq i \leq m$  because  $\gamma D$  is a closed clause)

Of course, all the names of the literals (i.e. the predicate symbols) in  $C$  are in  $D$  (as  $C \leq_s \gamma D$  and  $\gamma$  do not act on the predicate symbols).

To prove  $C \leq_s D$ , we must get back to  $D$ , or in other words, undo what was done by  $\gamma$ .

We thus define the operation:

$$s_i = t_i[c_1 | y_1, c_2 | y_2 \dots, c_n | y_n]$$

meaning: “ $s_i$  is the term obtained by replacing in  $t_i$  constant  $c_i$  by variable  $y_i$  ( $1 \leq i \leq n$ )”.

Consider the substitution:

$$\delta = \{x_1 \leftarrow s_1, x_2 \leftarrow s_2, \dots, x_m \leftarrow s_m\}$$

and let  $\Omega = [c_1 | y_1, c_2 | y_2 \dots, c_n | y_n]$  ( $\Omega$  is not a substitution; hence, the different notation)

$$\text{By construction: } \Omega(\gamma D) = D$$

$$\text{and also by construction } \delta C \subseteq D.$$

As a small example of these constructions, let (as usual  $x, y$  denote variables and  $a, b$  denote constants):

$$C : P(x) \vee Q(f(x)), \text{ hence, } Var(C) = \{x\}$$

$$D : P(f(y)) \vee Q(f(f(y))) \vee R(a), \text{ hence } Var(D) = \{y\}$$

$$\gamma = \{y \leftarrow b\}$$

$$\text{hence } \gamma D = P(f(b)) \vee Q(f(f(b))) \vee R(a)$$

$$(t_1 = t_2 = f(b))$$

$$\Omega = [b | y]$$

$$(s_1 = s_2 = f(y))$$

$$\delta = \{x \leftarrow f(y)\}$$

$$\delta C = P(f(y)) \vee Q(f(f(y))) (\subseteq D)$$

hence  $C \leq_s D$ .

Property 3 allows us to replace the study of  $C \leq_s D$  by the study of  $C \leq_s D_f$ , where  $D_f$  is a closed clause.

We use this property to answer the question.

**a)**

The clauses are disjunctions of literals with all variables quantified *universally*.

If  $C \leq_s D$ , then there exists a closed substitution that transforms  $C$  into a subclause of the closed clause  $D$ .

Every model of  $C$  evaluates  $C$  to **T** on all elements of the domain, *in particular*, on the elements denoted by the closed terms that replaced the variables of  $C$  (functional symbols are associated to *total* functions, see definition 5.6).

Hence, every model of  $C$  is a model of  $D$  and

if  $C \leq_s D$ , then  $C \models D$ .

**b)**

To answer question (b) it suffices to recall the definition of a subsumption and note that there exists an algorithm (UNIFICATION) that permits us to solve sets of equations on terms.

An (inefficient!) decision procedure could be that of Figure 12.1<sup>2</sup>:

We consider the input clauses, say  $C$  and  $D$ , as sets of literals:

$$C = \{K_1(\overline{s_1}), K_2(\overline{s_2}), \dots, K_m(\overline{s_m})\}$$

$$D = \{L_1(\overline{t_1}), L_2(\overline{t_2}), \dots, L_n(\overline{t_n})\}$$

$D$ : closed clause, meaning that the occurrence test (rule 7 of algorithm UNIFICATION is not necessary (see exercise 4.1 d)).

We note  $\mathcal{P}_{K_i}(D)$ , the set of literals of  $D$  whose names (predicate symbols) are the same as the name of  $K_i$  (of course, every literal name in  $C$  is a literal name in  $D$ , otherwise, the negative answer to the subsumption test is immediate).  $\square$

---

<sup>2</sup> This is a specification destined simply to show that a decision procedure exists, and is in no case intended to be implemented.

```

algorithm C SUBS D ?
input:  $C = \{K_1(\overline{s_1}), K_2(\overline{s_2}), \dots, K_m(\overline{s_m})\}$  and
          $D = \{L_1(\overline{t_1}), L_2(\overline{t_2}), \dots, L_n(\overline{t_n})\}$  ( $D$ : closed)
output: either YES and a substitution  $\sigma$ 
         or NO
begin
     $i \leftarrow 1$ ;
    fail  $\leftarrow \mathbf{F}$ 
     $\sigma \leftarrow \emptyset$     %  $\sigma$  : substitution to construct
    while  $i \leq m$  and  $\neg$  failure
    do
    • if  $\{\overline{s_i} \doteq \overline{t_j}\} \cup \sigma$  has a solution for a choice in  $\mathcal{P}_{K_i}(D)$ 
      % i.e. for the  $K_i \in C$  there exists  $L_j \in D$  s.t.  $(K_i(\overline{s_i}) \equiv L_j(\overline{t_j}))$ 
    • Memorize the other possible choices in  $\mathcal{P}_{K_i}(D)$ 
      % (for backtracking)
      then
         $\sigma \leftarrow \sigma \cup \{\overline{s_i} \doteq \overline{t_j}\}$ 
         $i \leftarrow i + 1$ 
      else
        if backtrack possible
        then  $\sigma \leftarrow \sigma \setminus$  “incorrect choices”
        else failure  $\leftarrow \mathbf{T}$ 
    enddo
    if failure then return NO
    else return YES and  $\sigma$ 
end

```

Figure 12.1. Subsumption algorithm for clauses

EXERCISE 9.1.– As we have to prove validity, we negate the corresponding formula

$$\begin{array}{ll}
 1) & \neg[\forall x \forall y (x = y \Rightarrow y = x)] \quad \checkmark \\
 & \downarrow \\
 2) & \exists x \exists y. \neg(x = y \Rightarrow y = x) \quad (1), \checkmark \\
 & \downarrow \\
 3) & \neg(a = b \Rightarrow b = a) \quad (2), x \leftarrow a, y \leftarrow b \checkmark \\
 & \downarrow \\
 4) & a = b \quad (3) \\
 5) & \neg(b = a) \quad (3) \\
 & \downarrow \\
 6) & \neg(b = b) \quad (4), (5), R_2^- \\
 & \quad \times \quad (6), R_1^-
 \end{array}$$

□

EXERCISE 9.2.–

As we have to prove validity, we negate the corresponding formula:

- 1)  $\neg[\forall x\forall y\forall z((x = y) \wedge (y = z) \Rightarrow (x = z))]$   $\checkmark$
- $\downarrow$
- 2)  $\exists x\exists y\exists z.\neg((x = y) \wedge (y = z) \Rightarrow (x = z))$  (1),  $\checkmark$
- $\downarrow$
- 3)  $\neg((a = b) \wedge (b = c) \Rightarrow (a = c))$  (2),  $x \leftarrow a, y \leftarrow b, z \leftarrow c, \checkmark$
- $\downarrow$
- 4)  $a = b$  (3)
- 5)  $b = c$  (3)
- 6)  $\neg(a = c)$  (3)
- $\downarrow$
- 7)  $a = c$  (4), (5),  $R_2^-$
- $\times$  (6), (7)

□

EXERCISE 9.4.–

% The closed world:

{  $1 \leq L1 \leq 4, 1 \leq L2 \leq 4, 1 \leq L3 \leq 4, 1 \leq L4 \leq 4, 1 \leq C1 \leq 4,$   
 $1 \leq C2 \leq 4, 1 \leq C3 \leq 4, 1 \leq C4 \leq 4,$

% The possible positions:

( $L1 = L2 + 1$  and  $C1 = C2 + 2$ ) or ( $L1 = L2 + 2$  and  $C1 = C2 + 1$ )  
 or ( $L1 = L2 - 1$  and  $C1 = C2 - 2$ ) or ( $L1 = L2 - 2$  and  $C1 = C2 - 1$ );  
 ( $L1 = L3 + 1$  and  $C1 = C3 + 2$ ) or ( $L1 = L3 + 2$  and  $C1 = C3 + 1$ )  
 or ( $L1 = L3 - 1$  and  $C1 = C3 - 2$ ) or ( $L1 = L3 - 2$  and  $C1 = C3 - 1$ );  
 $\vdots$   
 }

We also give a *schema* of propositional formulas that permits us to get the specification in PL of the  $n$ -queens problem for *all*  $n \in \mathbb{N}$ .

The proposition schemas  $P_{i,j}$  mean: *a queen is at the intersection of line  $i$  and column  $j$ .*

$$\bigwedge_{i,j=1}^n \neg [P_{i,j} \wedge ( \bigvee_{k=1; k \neq j}^n P_{i,k} \vee \bigvee_{l=1; l \neq i}^n P_{l,j} \vee \bigvee_{q,r=1; i-j \neq q-r}^n P_{q,r} \vee \bigvee_{s,t=1; i+j \neq s+t}^n P_{s,t} )]$$

or the equivalent schema:

$$\bigwedge_{i,j=1}^n [P_{i,j} \Rightarrow \neg ( \bigvee_{k=1; k \neq j}^n P_{i,k} \vee \bigvee_{l=1; l \neq i}^n P_{l,j} \vee \bigvee_{q,r=1; i-j \neq q-r}^n P_{q,r} \vee \bigvee_{s,t=1; i+j \neq s+t}^n P_{s,t} )]$$

where the disjuncts of the schema have the following meaning:

$\bigvee_{k=1; k \neq j}^n P_{i,k}$ : queen on a square of the same line;

$\bigvee_{l=1; l \neq i}^n P_{l,j}$ : queen on a square of the same column;

$\bigvee_{q,r=1; i-j \neq q-r}^n P_{q,r}$ : queen on a square of the same diagonal NE-SW going through  $i, j$ ;

$\bigvee_{s,t=1; i+j \neq s+t}^n P_{s,t}$ : queen on a square of the diagonal NW-SE going through  $i, j$ .  $\square$

EXERCISE 9.6.–  $R$  is a well-founded relation, meaning that there are no infinite sequences  $\{a_i\}_{i \in \mathbb{N}}$  such that  $R(a_1, a_0), R(a_2, a_1), \dots, R(a_{n+1}, a_n), \dots$

This is explained by taking:

$P \mapsto$  set of elements ( $\subseteq D$ )

$P(x) \mapsto x \in P$

The formula is interpreted as follows.

Every non-empty subset of the domain  $[\forall P(\exists x P(x) \dots)]$  contains an element  $[\exists z P(z) \dots]$ , such that there is no other element in the subset that is in relation with it  $[\dots \forall y (P(y) \Rightarrow \neg R(y, z) \dots)]$ .

b)

Connected graph

This is explained by taking

$C \mapsto$  set of nodes ( $\subseteq D$ )

$$C(x) \mapsto x \in C$$

$$A(x, y) \mapsto \text{there is an edge from } x \text{ to } y$$

The formula is then interpreted as follows:

For every non-empty subset of nodes [  $\forall C((\exists x C(x) \dots)$  ], if  $x$  is in the subset [  $\dots C(x) \dots$  ] and if when there is an edge from  $x$  to  $y$ , then  $y$  is also in this subset [  $\dots \wedge A(x, y) \Rightarrow C(y) \dots$  ] and all the nodes in the graph are in this subset [  $\dots \forall z C(z) \dots$  ].

c)

**Non-connected graph with connected components**

This is explained by taking:

$$C \mapsto \text{set of nodes } (\subseteq D)$$

$$C(x) \mapsto x \in C$$

$$A(u, v) \mapsto \text{there is an edge from } u \text{ to } v$$

The formula is then interpreted as follows.

There exists a non-empty subset of nodes of the graph [  $\forall C((\exists x C(x) \dots)$  ], which does not contain all the nodes in the graph [  $\dots \exists y \dots \neg C(y) \dots$  (disconnected part) ], and such that if a node belongs to this subset [  $\dots C(u) \dots$  ] and there is an edge from this node to another one [  $\dots A(u, v) \dots$  ], then the latter is also in the subset [  $\dots C(v) \dots$  (connected part) ].  $\square$

EXERCISE 10.1.–

a)

$C = A \cup B$ , by definition:

$$\mu_C = \max \{ \mu_A, \mu_B \}$$

$$\mu_C \geq \mu_A \text{ and } \mu_C \geq \mu_B$$

by definition:

$$1) A \subseteq C \text{ and } B \subseteq C$$



Let  $D$  such that:

$$2) A \subseteq D \text{ and } B \subseteq D$$

then:

$$\mu_D \geq \mu_A \text{ and } \mu_D \geq \mu_B$$

i.e.:

$$\mu_D = \max \{ \mu_A, \mu_B \} = \mu_C$$

by definition:

$$3) D = C$$

From (1), (2) and (3) we conclude that  $C$  is the smallest set containing  $A$  and  $B$ .

b)

$C = A \cap B$ , by definition:

$$\mu_C = \min \{ \mu_A, \mu_B \}$$

$$\mu_C \leq \mu_A \text{ and } \mu_C \leq \mu_B$$

by definition:

$$4) C \subseteq A \text{ and } C \subseteq B$$

Let  $D$  such that:

$$5) D \subseteq A \text{ and } D \subseteq B$$

then:

$$\mu_D \leq \mu_A \text{ and } \mu_D \leq \mu_B$$

i.e.:

$$\mu_D = \min \{ \mu_A, \mu_B \} = \mu_C$$

by definition:

$$6) D = C$$

From (4), (5), and (6), we conclude that  $C$  is the greatest set contained in  $A$  and  $B$ .

c)

(For every  $x$ ) we arrange in decreasing order  $\mu_A, \mu_B, \mu_C$

By transitivity of order relations:

$$\max\{\mu_i, \max\{\mu_j, \mu_k\}\} = \max\{\max\{\mu_i, \mu_j\}, \mu_k\}$$

where  $i, j, k \in \{A, B, C\}$

d)

similar (with *min*)

e)

There are two possible cases

$$e1) \mu_A \leq \mu_B$$

We verify easily that:

$$\mu_{\overline{(A \cup B)}} = 1 - \max\{\mu_A, \mu_B\} = 1 - \mu_B = \min\{1 - \mu_A, 1 - \mu_B\} = \mu_{\overline{A} \cap \overline{B}}$$

$$e2) \mu_B \leq \mu_A$$

Similar.

f)

Analogous to (e), by verifying:

$$1 - \min\{\mu_A, \mu_B\} = \max\{1 - \mu_A, 1 - \mu_B\}$$

g)

The property to verify here is:

$$\max\{\mu_C, \min\{\mu_A, \mu_B\}\} = \min\{\max\{\mu_C, \mu_A\}, \max\{\mu_C, \mu_B\}\}$$

which must be verified for the 6 (= 3!) possible cases:

$$g1) \mu_A \leq \mu_B \leq \mu_C$$

$$\text{g2) } \mu_A \leq \mu_C \leq \mu_B$$

$$\text{g3) } \mu_B \leq \mu_A \leq \mu_C$$

$$\text{g4) } \mu_B \leq \mu_C \leq \mu_A$$

$$\text{g5) } \mu_C \leq \mu_A \leq \mu_B$$

$$\text{g6) } \mu_C \leq \mu_B \leq \mu_A$$

We verify for (g1)

$$\max \{ \mu_C, \min \{ \mu_A, \mu_B \} \} = \mu_C$$

$$\max \{ \mu_C, \mu_A \} = \mu_C$$

$$\max \{ \mu_C, \mu_B \} = \mu_C$$

$$\min \{ \mu_C, \mu_C \} = \mu_C$$

h) Analogous to (g) by verifying:

$$\min \{ \mu_C, \max \{ \mu_A, \mu_B \} \} = \max \{ \min \{ \mu_C, \mu_A \}, \min \{ \mu_C, \mu_B \} \}. \quad \square$$

EXERCISE 10.2.– For example:

$$\mu_{x \ll y}(0, 2) = 0.1$$

$$\mu_{x \ll y}(5, 150) = 0.8$$

$$\mu_{x \ll y}(0, 1000) = 0.99$$

⋮

□

EXERCISE 10.3.–

a)

Let  $R \subseteq A \times B$  and  $S \subseteq B' \times C$

If  $B \cap B' = \emptyset$  then  $\mu_{R \circ S} = 0$  for every pair  $(x, y)$

If  $B \cap B' \neq \emptyset$ , then we must keep the pairs containing the  $z$ 's that are in  $B$  and in  $B'$ , i.e.:

$$\min\{\mu_R(x, z), \mu_S(z, y)\}$$

and regroup all these values:

$$\mu_{R \circ S} = \max_z \{\min\{\mu_R(x, z), \mu_S(z, y)\}\}$$

b)

$$\mu_R(x, x) \approx 1$$

c)

$$\text{if } \mu_R(x, y) \approx 1 \text{ then } \mu_S(y, x) \approx 1$$

d)

In classical logic: if  $xRy$  and  $yRx$ , then  $x = y$

We can propose:

$$\text{if } \mu_R(x, y) \approx 1, \text{ then } \mu_R(y, x) \approx 0$$

or:

$$\text{if } \mu_R(x, y) > 0, \text{ then } \mu_R(y, x) = 0$$

e)

In classical logic: if  $xRy$  and  $yRz$ , then  $xRz$

$$\mu_R(x, z) \geq \min\{\mu_R(x, y), \mu_R(y, z)\}$$

The idea is that  $x$  will be related to  $z$  at least the same degree as  $x$  and  $y$  are related (which corresponds to *min*).  $\square$

EXERCISE 10.4.– Yes:

$S \xrightarrow{1} A \xrightarrow{4} A$  and  $B \xrightarrow{3} B$  and  $B \xrightarrow{6} \text{not } C$  and  $B \xrightarrow{8} \text{not } D$  and  $B \xrightarrow{10} \text{not very } D$  and  $B \xrightarrow{12} \text{not very true}$  and  $B \xrightarrow{6} \text{not very true}$  and  $\text{not } C \xrightarrow{9} \text{not very true}$  and  $\text{not } E \xrightarrow{11} \text{not very true}$  and  $\text{not very } E \xrightarrow{13} \text{not very true}$  and  $\text{not very false}$

The semantics:

$$\frac{S \xrightarrow{1} A \xrightarrow{4} A \cap B \xrightarrow{3} B \cap B \xrightarrow{6} \overline{C} \cap B \xrightarrow{8} \overline{D} \cap B \xrightarrow{10} \overline{D^2} \cap B \xrightarrow{12}}{\text{true}^2 \cap B \xrightarrow{6} \text{true}^2 \cap \overline{C} \xrightarrow{9} \text{true}^2 \cap \overline{E} \xrightarrow{11} \text{true}^2 \cap \overline{E^2} \xrightarrow{13} \text{true}^2 \cap \text{false}^2} \square$$

## EXERCISE 10.5.–

a)

- |   |   |
|---|---|
| 1) $\Box \neg P \Rightarrow \neg P$           | T – axiom $P \leftarrow \neg P$             |
| 2) $\neg \neg P \Rightarrow \neg \Box \neg P$ | (1) and contrapositive                      |
| 3) $P \Rightarrow \neg \Box \neg P$           | in (2) propositional equiv                  |
| 4) $P \Rightarrow \Diamond P$                 | $\Diamond P : \text{def } \neg \Box \neg P$ |

b)

- |  |                    |
|--|--------------------|
| 1) $A \Rightarrow B$   | hyp.               |
| 2) $\Box(A \Rightarrow B)$   | (1), Necessitation |
| 3) $\Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$ | axiom K            |
| 4) $\Box A \Rightarrow \Box B$                                     | (2), (3), MP       |

c)

- |   |  |
|---|--|
| 1) $\Box \neg P \Rightarrow \Box \Box \neg P$           | S4 – axiom $P \leftarrow \neg P$                 |
| 2) $\neg \Box \Box \neg P \Rightarrow \neg \Box \neg P$ | (1) contrapositive                               |
| 3) $\Diamond \neg \Box \neg P \Rightarrow \Diamond P$   | $\Diamond P : \text{def } \neg \Box \neg P$      |
| 4) $\Diamond \Diamond P \Rightarrow \Diamond P$         | (3), $\Diamond P : \text{def } \neg \Box \neg P$ |

d)

- |  |  |
|--|--|
| 1) $\Box(\neg Q \Rightarrow P) \Rightarrow (\Box \neg Q \Rightarrow \Box P)$ | axiom K (hence in S5)<br>$P \leftarrow \neg Q, Q \leftarrow P$ |
| 2) $\Box(Q \vee P) \Rightarrow (\neg \Box \neg Q \vee \Box P)$               | $Q \Rightarrow P \text{ equiv } \neg Q \vee P$                 |
| 3) $\Box(P \vee Q) \Rightarrow (\Box P \vee \Diamond Q)$                     | $\Diamond Q : \text{def } \neg \Box \neg Q$                    |

e)

- |  |   |
|--|---|
| 1) $A \wedge B \Rightarrow A$  | tautology PL  |
| 2) $A \wedge B \Rightarrow B$  | tautology PL  |
| 3) $\Box(A \wedge B \Rightarrow A)$  | (1), Necessitation  |
| 4) $\Box(A \wedge B \Rightarrow B)$  | (2), Necessitation  |
| 5) $\Box(A \wedge B \Rightarrow A)$<br>$\Rightarrow (\Box(A \wedge B) \Rightarrow \Box A)$ | axiom K   |
| 6) $\Box(A \wedge B \Rightarrow B)$<br>$\Rightarrow (\Box(A \wedge B) \Rightarrow \Box B)$ | axiom K   |
| 7) $\Box(A \wedge B) \Rightarrow \Box A$   | (3), (5) MP   |
| 8) $\Box(A \wedge B) \Rightarrow \Box B$   | (4), (6) MP   |
| 9) $\Box(A \wedge B) \Rightarrow \Box A \wedge \Box B$                                     | PL, if $A \Rightarrow B$ taut. and $A \Rightarrow C$ taut.<br>then $A \Rightarrow B \wedge C$ taut. |

f)

- |  |  |
|--|--|
| 1) $\Box(B \Rightarrow (A \wedge B)) \Rightarrow (\Box B \Rightarrow \Box(A \wedge B))$                                      | axiom K  |
| 2) $A \Rightarrow (B \Rightarrow (A \wedge B))$  | tautology PL   |
| 3) $\Box(A \Rightarrow (B \Rightarrow (A \wedge B)))$  | (2), Necessitation   |
| 4) $\Box(A \Rightarrow (B \Rightarrow (A \wedge B)))$<br>$\Rightarrow (\Box A \Rightarrow \Box(B \Rightarrow (A \wedge B)))$ | Axiom K  |
| 5) $\Box A \Rightarrow \Box(B \Rightarrow (A \wedge B))$   | (3), (4), MP   |
| 6) $\Box A \Rightarrow (\Box B \Rightarrow \Box(A \wedge B))$  | (5), (1) transitivity PL   |
| 7) $\Box A \wedge \Box B \Rightarrow \Box(A \wedge B)$   | LP: if $A \Rightarrow (B \Rightarrow C)$<br>taut. : $A \wedge B \Rightarrow C$ taut. |

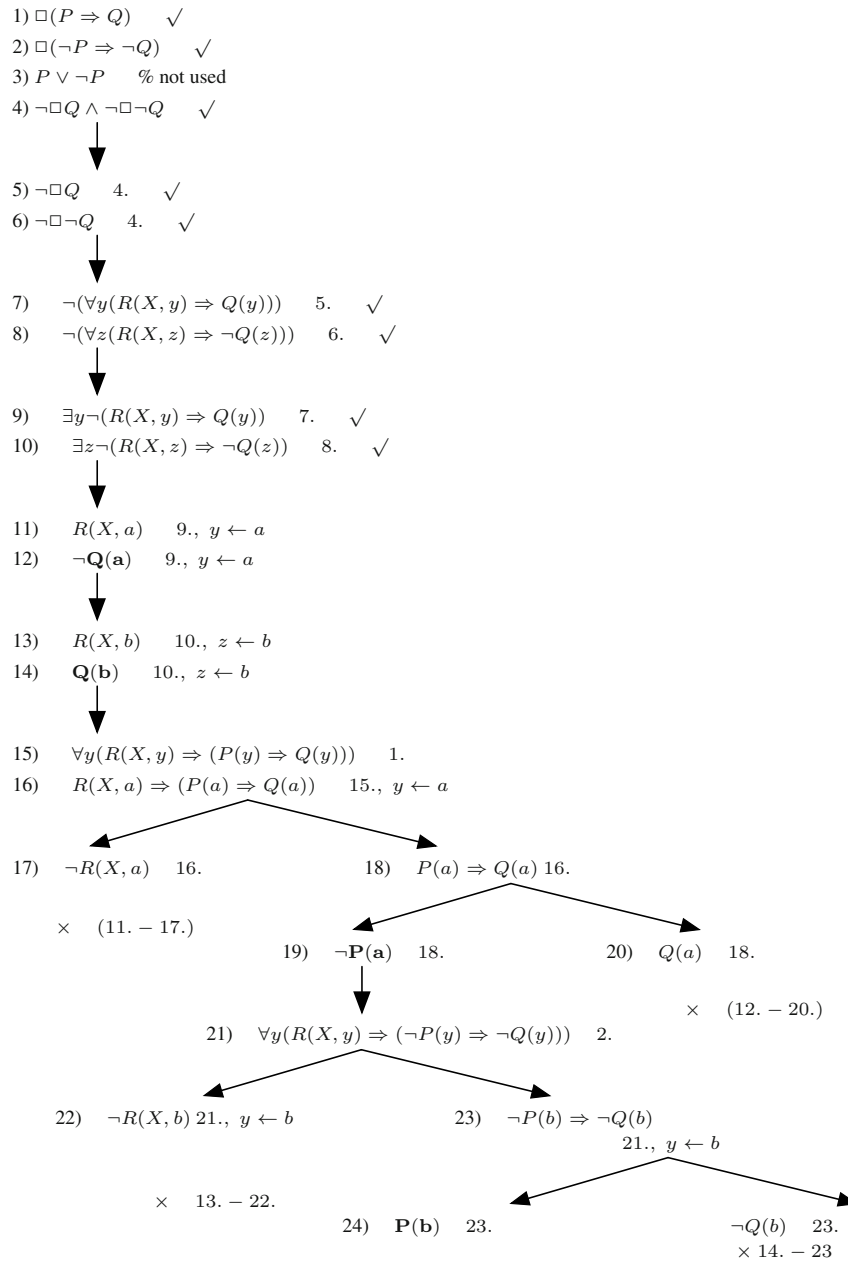
g)

- |  |   |
|--|---|
| 1) $A \Rightarrow (B \wedge C \Rightarrow A \wedge B)$   | tautology PL  |
| 2) $P \Rightarrow (\underbrace{\Box P \wedge \Box \Box P}_{\Box P \wedge \Box \Box P} \Rightarrow P \wedge \Box P)$                                    | (1) $A \leftarrow P, B \leftarrow \Box P, C \leftarrow \Box \Box P$           |
| 3) $\Box(P \wedge \Box P) \Leftrightarrow (\underbrace{\Box P \wedge \Box \Box P}_{\Box P \wedge \Box \Box P})$  | (e), (f)  |
| 4) $P \Rightarrow (\Box(P \wedge \Box P) \Rightarrow P \wedge \Box P)$   | (3) in (2)  |
| 5) $\Box(P \Rightarrow (\Box(P \wedge \Box P) \Rightarrow P \wedge \Box P))$   | (4), Necessitation  |
| 6) $\Box P \Rightarrow \underbrace{\Box(\Box(P \wedge \Box P) \Rightarrow P \wedge \Box P)}_{\Box(\Box(P \wedge \Box P) \Rightarrow P \wedge \Box P)}$ | (5), axiom K (hence of G), MP   |
| 7) $\underbrace{\Box(\Box(P \wedge \Box P) \Rightarrow (P \wedge \Box P))}_{\Rightarrow \Box(P \wedge \Box P)}$  | axiom G : $P \leftarrow P \wedge \Box P$                                      |
| 8) $\Box P \Rightarrow \Box(P \wedge \Box P)$  | (6), (7) transitivity PL  |
| 9) $\Box P \Rightarrow \Box P \wedge \Box \Box P$  | (e), (8) transitivity PL  |
| 10) $\Box P \Rightarrow \Box \Box P$   | (9), LP : if $A \Rightarrow B \wedge C$ taut.<br>then $A \Rightarrow C$ taut. |

h)

- |  |  |
|--|--|
| 1) $\Box A \Rightarrow \Box A \vee \Box B$       | $P \Rightarrow P \vee Q$ PL tautology, $P \leftarrow \Box A,$<br>$Q \leftarrow \Box B$ |
| 2) $\Box B \Rightarrow \Box A \vee \Box B$       | idem   |
| 3) $\Box(\Box A \Rightarrow \Box A \vee \Box B)$ | (1), Necessitation   |
| 4) $\Box(\Box B \Rightarrow \Box A \vee \Box B)$ | (2), Necessitation   |
| 5) $\Box A \Rightarrow \Box \Box A$              | Axiom S4   |
| 6) $\Box B \Rightarrow \Box \Box B$              | Axiom S4   |





In bold ( $\{\neg Q(a), Q(b), \neg P(a), P(b)\}$ ), we can read a counter example of this formalization of the argument (reasoning) of the sea battle.  $\square$

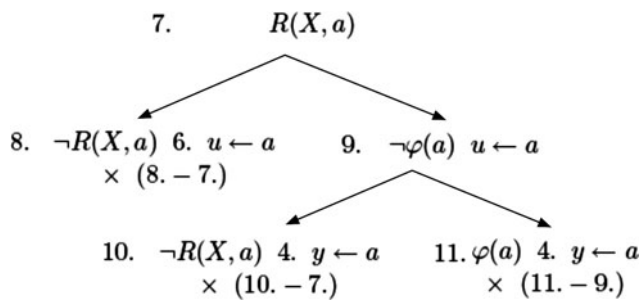


EXERCISE 10.7.–

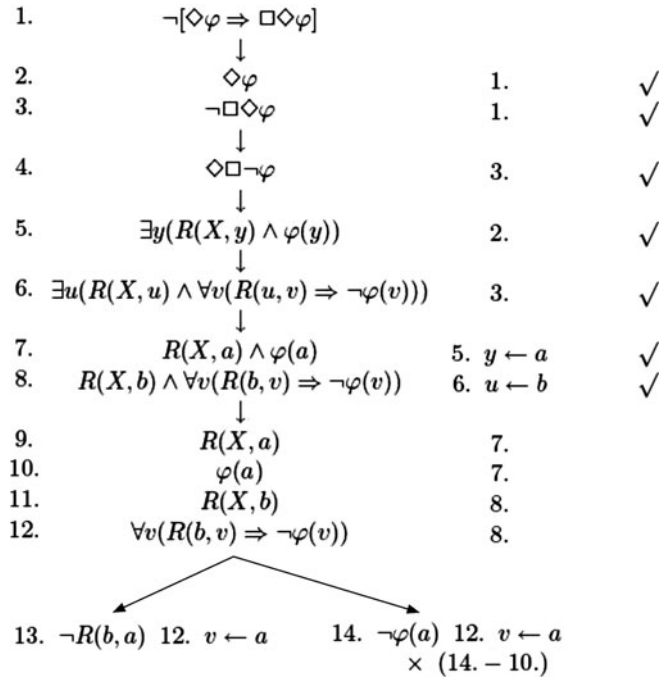
a)

1.  $\neg[\Box\varphi \Rightarrow \Diamond\varphi]$
- $\downarrow$
2.  $\Box\varphi$             1.  $\checkmark$
3.  $\neg\Diamond\varphi$         1.  $\checkmark$
- $\downarrow$
4.  $\forall y(R(X, y) \Rightarrow \varphi(y))$     2.
5.  $\neg\exists u(R(X, u) \wedge \varphi(u))$     3.  $\checkmark$
- $\downarrow$
6.  $\forall u\neg(R(X, u) \wedge \varphi(u))$     5.

If  $R$  is total, i.e.  $\forall x\exists yR(x, y)$ , we have  $R(X, a)$  (by assigning  $x \leftarrow X, y \leftarrow a$ ) and we can graft the following tree:

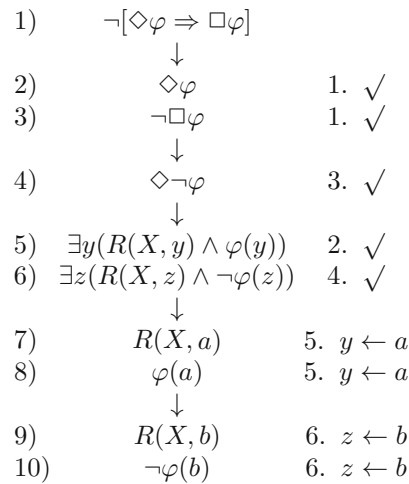


b)



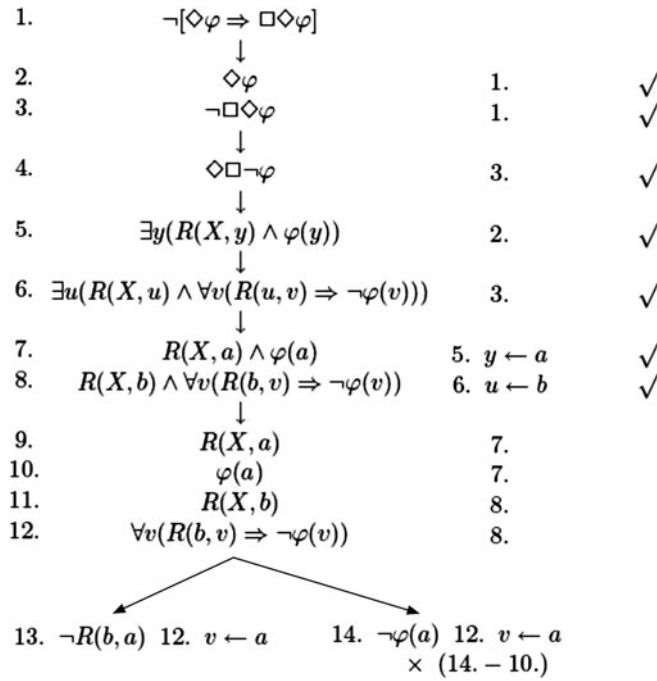
If  $R$  is **Euclidean**, i.e.  $\forall x\forall y\forall z.R(x, y) \wedge R(x, z) \Rightarrow R(y, z)$ , then we have  $R(X, b) \wedge R(X, a) \Rightarrow R(b, a)$  and we can also close the left-hand branch.

c)



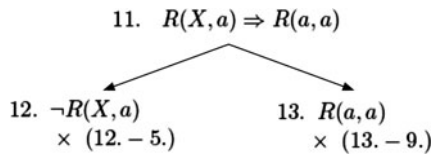
If  $R$  is functional (“unique”), i.e.  $\forall x\forall y\forall z.R(x, y) \wedge R(x, z) \Rightarrow y = z$ ,  
 then we have  $a = b$  (7 – 9) and the tree can be closed (8 – 10)

d)



If  $R$  is shift-reflexive, i.e.  $\forall x\forall y.R(x, y) \Rightarrow R(y, y)$ , then we have  $R(X, a) \Rightarrow R(a, a)$ .

We can thus close the tree by grafting to the open branch the tree:



e)

1.	$\neg(\Diamond\Box P \Rightarrow \Box\Diamond P)$			
	↓			
2.	$\Diamond\Box P$	1.	✓	
3.	$\neg\Box\Diamond P$	1.	✓	
	↓			
4.	$\Diamond\Box\neg P$	3.		
	↓			
5.	$\exists y(R(X, y) \wedge \forall u(R(y, u) \Rightarrow P(u)))$	2.	✓	
6.	$\exists v(R(X, v) \wedge \forall z(R(v, z) \Rightarrow \neg P(z)))$	4.	✓	
	↓			
7.	$R(X, a) \wedge \forall u(R(a, u) \Rightarrow P(u))$	5.	$y \leftarrow a$	✓
8.	$R(X, b) \wedge \forall z(R(b, z) \Rightarrow \neg P(z))$	6.	$v \leftarrow b$	✓
	↓			
9.	$R(X, a)$	7.		
10.	$R(X, b)$	8.		
11.	$\forall u(R(a, u) \Rightarrow P(u))$	7.		
12.	$\forall z(R(b, z) \Rightarrow \neg P(z))$	8.		
-----				
12'.	$\exists w(R(a, w) \wedge R(b, w))$	9., 10. and $R$ : directed	✓	
	↓			
12''	$R(a, c)$	12'	$w \leftarrow c$	
12'''	$R(b, c)$	12'	$w \leftarrow c$	
-----				
13.	$\neg R(a, c)$	11. $u \leftarrow c$	×	
	↙ ↘			
14.	$P(c)$	11. $u \leftarrow c$		
	↙ ↘			
	$\neg R(b, c)$	12. $z \leftarrow c$	×	
	↙ ↘			
	$\neg P(c)$	12. $z \leftarrow c$	×	

Line 12' corresponds to  $R$  confluent (or convergent), i.e.  $\forall x\forall y\forall z.R(x, y) \wedge R(x, z) \Rightarrow \exists u(R(y, u) \wedge R(z, u))$  □

EXERCISE 10.8.– If we keep the semantics of  $\Box\varphi$  in definition 10.9:

$O\varphi$ :  $\varphi$  is compulsory  $O\varphi :^{def} \Box\varphi$

$F\varphi$ :  $\varphi$  is forbidden  $F\varphi :^{def} O\neg\varphi$

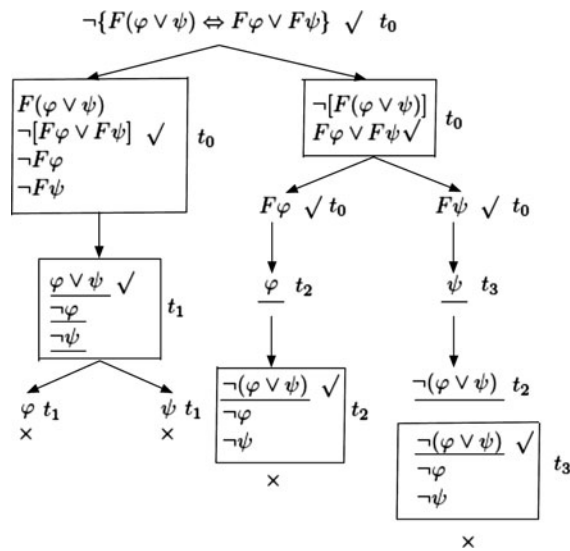
$P\varphi$ :  $\varphi$  is allowed  $P\varphi :^{def} \neg O\neg\varphi$

$E\varphi$ :  $\varphi$  is optional  $E\varphi :^{def} \neg O\varphi$ . □

EXERCISE 10.9.–

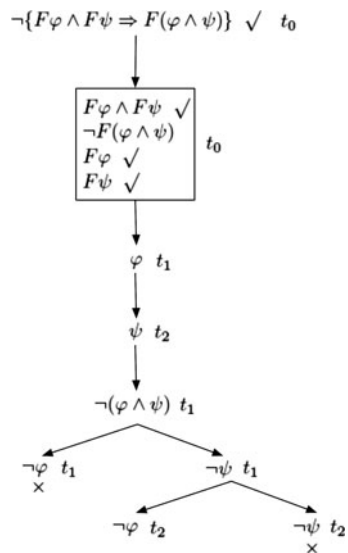
a)

The underlined formulas at time  $t_i$  are those produced by the application of a rule at time  $t_j$  ( $j < i$ ).



The tree is closed, hence the formula is valid.

b)

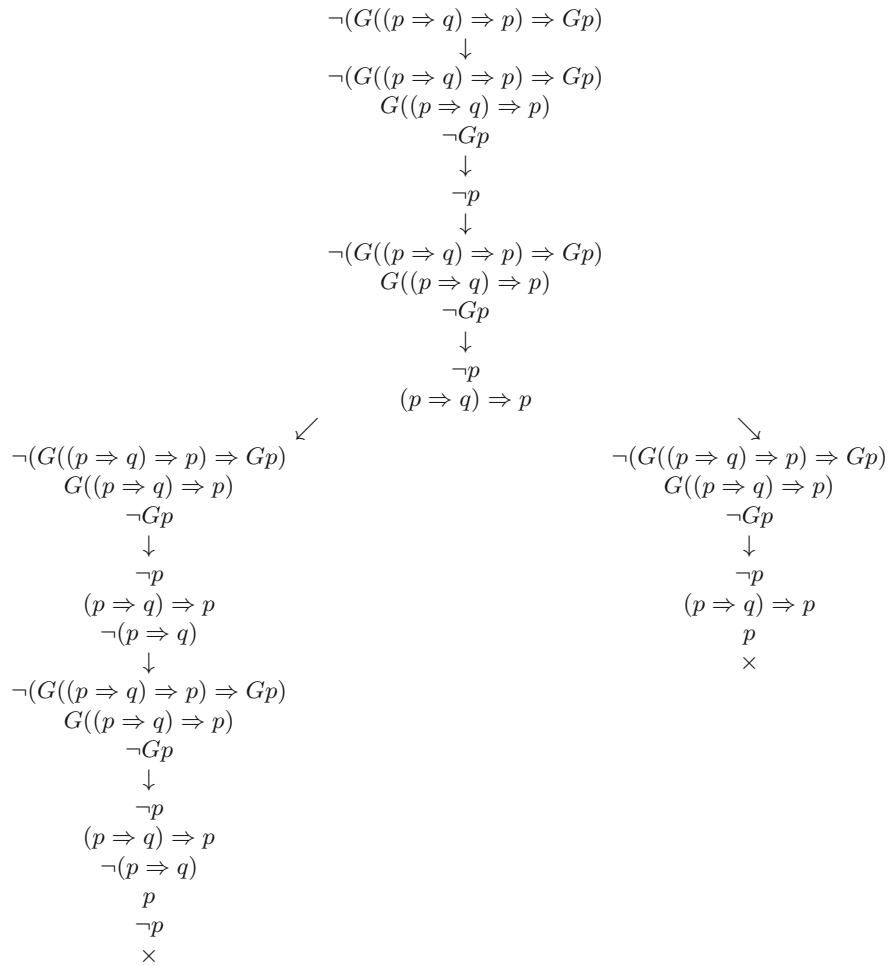


The formula is not valid. Counter example:

$$\{\varphi[t_1], \psi[t_2], \neg\psi[t_1], \neg\varphi[t_2], \dots\}$$

which could translate into: “if it will rain and the weather will be cold, that does not mean it will rain and the weather will be cold at the same time!”.  $\square$

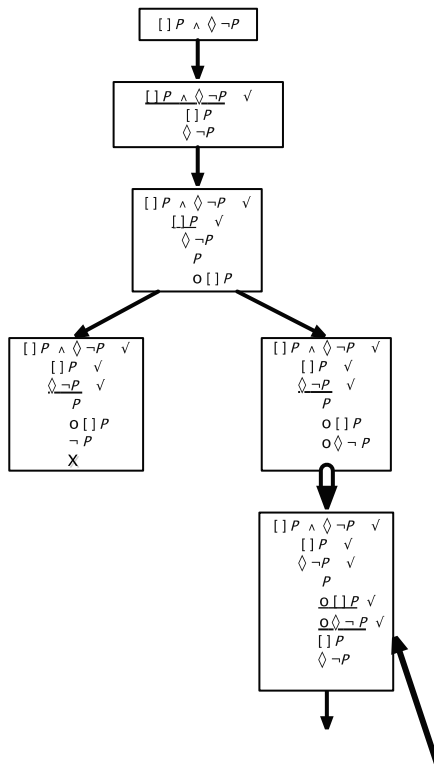
EXERCISE 10.10.– We can find representations in the literature similar to the representation below. It is explicit enough. The reader can name the instants for each group of formulas.

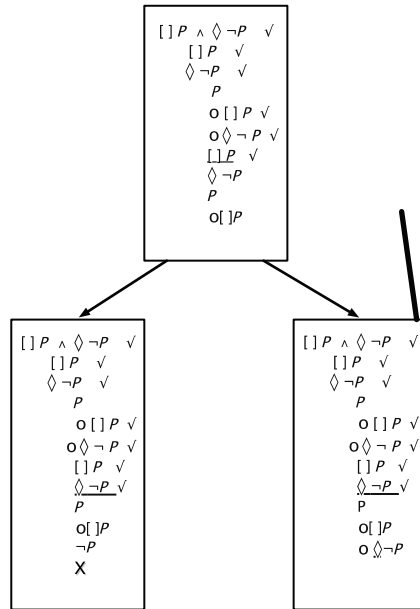


$\square$

EXERCISE 10.11.— The rule corresponding to  $\Box f$ :

$\Box f$   
 $\downarrow$   
 $f$   
 $\circ \Box f$





The underlined formulas are those that *were developed* in the parent node (rectangle). □



## Bibliography

- [BAR 77] BARWISE J., “An introduction to first-order logic”, in BARWISE J. *et al.* (eds), *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, Chapter A.1, pp. 5–46, North Holland, Amsterdam, 1977.
- [BEN 88] VAN BENTHEM J.F.A.K., *A Manual of Intensional Logic*, volume 1 of *Lecture Notes*, CSLI Publications, Menlo Park, 1988, 2nd ed., revised and expanded.
- [BOO 89] BOOLOS G.S., JEFFREY R.C., *Computability and Logic*, Cambridge University Press, Cambridge, 1989.
- [EBB 84] EBBINGHAUS H.D., FLUM J., THOMAS W., *Mathematical Logic*, Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1984.
- [END 72] ENDERTON H.B., *A Mathematical Introduction to Logic*, Academic Press, New York, 1972.
- [FAG 95] FAGIN R., HALPERN J.Y., MOSES Y., VARDI M.Y., *Reasoning About Knowledge*, MIT Press, Cambridge, 1995.
- [FIT 90] FITTING M., *First-Order Logic and Automated Theorem Proving*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1990.
- [GOL 93] GOLDBLATT R., *Mathematics of Modality*, volume 43 of *CSLI Lecture Notes*, CSLI Publications, Menlo Park, 1993.
- [KLE 71] KLEENE S.C., *Logique Mathématique, Épistémologie*, Librairie Armand Colin, Paris, 1971.
- [LEI 97] LEITSCH A., *The Resolution Calculus*, Texts in Theoretical Computer Science, Springer, Berlin, 1997.
- [MEN 64] MENDELSON E., *Introduction to Mathematical Logic*, The University Series in Undergraduate Mathematics, D. Van Nostrand Co., Toronto, 1964.
- [ROB 79] ROBINSON J.A., *Logic: Form and Function. The Mechanization of Deductive Reasoning*, The University Press Edinburgh, Edinburgh, 1979.
- [SMU 68] SMULLYAN R.M., *First-Order Logic*, Springer-Verlag, Berlin, 1968.
- [WOS 92] WOS L., OVERBEEK R., LUSK E., BOYLE J., *Automated Reasoning, Introduction and Applications*, McGraw-Hill, New York, 2nd ed., 1992.



## Index

### Symbols

- $C_G$ , 391
  - $D_G$ , 391
  - $E_G$ , 391
  - $K_i$ , 389
  - $K_i$  operator, 389
  - $L[\bar{x} \mid \bar{t}_{H(S)}]$ , 177
  - $L[t]$ , 301
  - $L[t]_u$ , 301
  - $L[u \leftarrow t]$ , 301
  - $L^c$ , 46
  - $\Box$ , 101
  - Propset* of a wff (or a set of wffs), 47
  - $\Leftrightarrow$ , 41
  - $\Rightarrow$ , 20, 41
  - $\perp$ , 20
  - $\mathcal{R}$  operator in FOL, 192
  - $\mathcal{R}$  in FOL, 192
  - $\doteq$ , 127
  - $\Downarrow$ , 44
  - $\exists$ , 134
  - $\forall$ , 134
  - $\leq_s$ , 267
  - $\mathcal{E}$ , 47
  - $\mathcal{E}(A, I)$ , 42
  - $\mathcal{L}_1$ , 133
  - $\mathcal{R}$ , 103
  - $\mathcal{R}$ -operator, 103
  - $\mathcal{S}_1$ , 78
  - $\mathcal{S}_{LIO}$ , 184
  - $\mid$ , 44
  - $\models$ , 43
  - $\models_{\mathcal{I}}$ , 43
  - $\neg$ , 41
  - $\omega$ -consistency, 85
  - $\prec$ , 118
  - $\vdash_{\mathcal{R}}$ , 104
  - $\vdash_{\mathcal{S}_R}$ , 104
  - $\vdash$ , 78
  - $\vee$ , 41
  - $\wedge$ , 41
  - (abstract) algebra, 145
  - (strongly) confluent relation, 377
  - modus ponens* (MP), 72, 79, 89
  - shift-reflexive relation, 509
  - 3-colourability, 113
- ### A
- abduction, 273
  - absolutely consistent, 84
  - abstract interpreter of LP, 221
  - accessibility relation, 358
  - adequate (connective set), 44
    - formal system, 84
  - algorithm for semantic tableaux (PL), 61
  - ancestor (of a node), 63
  - antisymmetry, 117
  - argumentation, 48
  - arity, 125, 133, 294
  - artificial intelligence, 245
  - associativity without  $=$ , 296
  - asymmetry, 118

- axiom, 75
  - (for constructivists), 13
  - (of a theory), 151
  - of choice, 11
  - (sequent calculus), 91
- axiomatic
  - structure, 75
  - system, 75
- axiomatico-deductive system, 74
- axiomatisation, 75
  - of equality, 293
- B**
- backward chaining, 51
- Berry paradox, 13
- binary resolution, 191
- body of a rule, 219
- Boolean algebra, 116
- bound
  - occurrence of a variable, 135
  - variable, 135
- branch, 63
- C**
- calculus, 75, 109, 110
- case analysis, 74
- chain (totally ordered set), 117
- choice (axiom of), 11
- clausal
  - form (FOL), 45, 187
  - transformation, 49
- clause, 46, 219
  - (FOL), 187
  - (PL), 46
- clauses parents, 191
- closed
  - branch, 56, 97
  - clause, 177
  - instance, 177
  - semantic tree, 97
  - substitution, 126
  - tableau, 56
  - terms, 125, 177
  - theory, 151
  - wff, 136
- closure
  - of an owff, 148
  - operation, 27
- cnf, 45
  - transformation, 49
- coherent, 47, 84
- common knowledge, 391
- commutativity without =, 296
- compactness
  - of FOL, 185
  - (of PL), 99
  - theorem (PL), 99
- complementary literal, 46
- complete, 84
- complete theory, 151
- completeness
  - of FOL, 183
  - (for refutation), 105
- completion (Clark), 239
- complexity
  - of a proof by resolution, 112
  - of the resolution method, 112
- computability and Horn clauses, 241
- conclusion, 75
- conditional formula, 113
- confluent relation, 377
  - confluent (convergent) relation, 510
- congruence relation, 119
- conjunct, 46
- conjunctive
  - clauses, 103
  - normal form, 45
- consensus, 103
- consequence operation, Tarski, 27
- consequence relation
  - (*entailment relation*), 28
  - (Tarski), 27
- consistent, 47, 84
  - (absolutely), 84
  - for negation, 84
- constant instance, 177
- constraint, 309
  - projection, 310
- constructive proof, 17
- constructivist proof, 19
- continuity, 381
- contradictory, 47

copy, 192  
 counter-model (counterexample), 47  
 counterexample (counter-model), 47  
   of a set of formulas, 47  
 cwff: closed wff, 136

**D**

Davis and Putnam, 92  
   algorithm, 94  
 decidable, 84  
   class (some), 205  
 decision procedure, 84  
 deduction, 77  
   theorem, 81, 332  
 deductive  
   inference, 260  
   system, 75  
 demodulation, 303, 304  
 density, 381  
 depth of a term, 126  
 descendant (of a node), 63  
 description logic, 272  
 diagonalization, 74  
 direct consequence, 75  
 directed graph, 63, 383  
 discovery of explanatory theories, 274  
 disjunct, 46  
 disjunctive normal form, 46  
 distributed knowledge, 391  
 dnf, 46  
   transformation, 49  
 domain of discourse, 144

**E**

e-frame, 389  
 E-unsatisfiable, 303  
 empty clause, 101  
 eq (Prolog), 237  
 equality, 291  
   axiomatization, 293  
 equivalence  
   of formal systems, 87  
   relation, 119  
 Euler circles, 114  
 evaluation, 47  
 execution control (*/*), 229

existential quantifier, 134  
 expressive power, 321  
 expressivity, 321  
 extension, 22  
 extensional  
   connective, 22, 329  
   language, 22

**F**

factor (of a clause), 191  
 factorization, 191  
 failure node, 97  
 finite  
   model (modal logics), 359  
   model property, 202  
   set, 322  
   trees, 123  
 finitely  
   controllable class, 202  
   satisfiable, 326  
 first-order  
   logic, 131, 133  
   structure, 144  
 flat terms, 243  
 FOL, 131  
   (first-order logic), 133  
   semantics, 146  
 formal  
   system, 74  
   system for FOL, 183  
   systems approach for modal logics, 360  
   theory, 74  
 forward chaining, 51  
 free occurrence of a variable, 135  
 free variables, 135, 148  
 Frege system, 87  
 fundamental theorem of arithmetic, 207  
 fuzzy logic, 337  
 fuzzy set, 342

**G**

Gödel's  
   completeness theorem, 183  
   incompleteness theorem, 92, 206  
   (proof), 208  
 generalization, 284, 285

generalized characteristic function, 342

Gentzen

(LK system), 90

system, 88

greatest lower bound, 118

## H

halting problem, 13

head of a rule, 219

Herbrand

base, 176, 177

instance, 177, 183

interpretation, 176, 177

terms, 177

theorem, 183

theorem (for clauses), 189

universe, 176

hierarchy (priority) of connectives, 40

Hilbert

(proof procedure), 109

system, 87

Horn clause, 113

(computability), 241

## I

implication, 20

incompleteness theorem, 206

(Gödel), 92

independence, 87

induction axiom, 76

(extensional version), 321

(intensional version), 321

induction

(proof by), 73

principle, 73

inductive inference, 278, 280

inference, 259

node, 97

rule, 75

infimum, 118

infinite

set, 323

tree, 63

initial sequent, 91

intension, 22

interpolant, 54

interpretation, 41, 46

of a formula, 47

of a set of formulas, 47

(in FOL), 146

intuitionistic proof, 19

irreflexive relation, 63, 165

irreflexivity, 118

## K

K (minimal logic), 360

K-modal frame, 358

König's lemma, 63

knowledge, 385

(definition), 388

modal logic, 389

Kripke, 356

model, 358

semantics, 356, 358

## L

Löwenheim–Skolem theorem, 179

language, 30

lattice, 116

least upper bound, 117

left-hand side of a rule, 219

Leibniz's law, 293

length of a clause, 46

liar's paradox, 8

Lindenbaum algebra, 119

linear

strategy, 198

term, 451

literal, 46

LK (Gentzen's system), 90

logic

(formal definition), 26

(informal definitions), 24

(multi-valued), 327

programming, 213, 219

logical

axiom, 87

connective, 40

consequence, 47

constant, 134, 273

- inference rule, 87
- symbols, 134
- LP abstract interpreter, 221
- M**
- matching, 127
- meta-language, 30
- MFOL (pure monadic class), 203
- MFOL<sup>=</sup> (monadic first-order logic), 203
- mgu: most general unifier, 127
- minimal logic (K), 360
- minimally unsatisfiable, 53, 108
- modal
  - frame, 358
  - logics, 353
- model, 47
  - (general notion), 137
  - intersection property, 113
  - of a set of formulas (FOL), 147
  - of a set of formulas (PL), 47
- monadic
  - class, 202, 203
  - class of FOL, 203
  - first-order logic, 203
- monotonicity, 28
- monotony, 309
- more general clause, 267
- MP (*modus ponens*), 79, 89
- multi-sorted structure, 150
- multi-valued logic, 327
- N**
- n-valid formula, 162
- n-validity, 162
- NAF (negation as failure), 232
- natural deduction system, 87, 88
- negation as failure (NAF), 232
- negative
  - clause, 46
  - FOL, 134
  - literal, 46
  - normal form, 45
- nnf, 45
- non-compactness of SOL, 325
- non-logical symbols, 134
- non-rational infinite trees, 124
- non-reflexive relation, 63
- non-standard model, 153
- non-standard model, 152
- nondeterminism, 107, 109
- normal form (Skolem), 176
- not (Prolog), 232
- O**
- ontologies, 4
- ontology, 272
- open
  - branch, 56, 97
  - semantic tree, 97
  - tableau, 56
  - wff, 136
- order relation, 117
- owff: open wff, 136
- P**
- p-valued logic, 327
- paradox, 8
- paramodulant, 302
- paramodulation, 300
- parent clause, 102
- partial interpretation, 46
  - of a formula, 47
- partially ordered set, 117, 118
  - alternate definition, 118
- PC (propositional logic), 39
- Peano's axioms, 76
- PFOL (pure first-order logic), 134
- pigeonhole principle, 112, 324
- PL (propositional logic), 39
- PL syntax, 40
- positive
  - clause, 46
  - literal, 46
- possible worlds semantics, 356
- predicate logic, 131
- predicative programming, 241
- premise, 29, 75
- prenex normal form, 169
- Presburger arithmetic, 69, 147
- principle of mathematical induction, 73
- proof, 64, 77
  - by cases, 74

- by diagonalisation, 74
  - by induction, 73
  - (in LK), 91
  - procedure, 110
  - system, 75
  - proper
    - axiom, 87
    - inference rule, 87
  - properties of the knowledge, 394
  - proposition, 39
    - (formal definition), 119
  - propositional logic (PL), 39
  - provable formula, 76
    - in a formal system, 76
  - pure
    - first-order logic, 134
    - literal, 51, 93, 405, 420
  - purity principle, 420, 432
- Q**
- quantified variable, 134
  - quotient algebra, 119
- R**
- rational infinite trees, 124
  - reasoning, 48
  - reductio ad absurdum, 73
  - reduction, 303
  - reflexivity, 117
  - refutation, 104
  - refutational completeness, 105, 106
    - (FOL resolution), 192
  - relational
    - programming, 241
    - system, 145
  - resolution
    - (in FOL), 190
    - (in PL), 101
    - method (in FOL), 190
    - (refutational completeness for FOL), 192
    - rule, 102
  - resolvent, 102
  - rewriting, 303
  - right-hand side of a rule, 219
  - rule, 219
  - Russell's paradox, 6, 10
- S**
- SAT problem, 92
  - satisfaction, 139
    - relation (in FOL), 146
  - satisfiability
    - by resolution, 192, 438
    - validity (in modal logic), 358
  - satisfiable, 47
    - formula in an e-frame, 390
  - saturation, 438
  - scope of quantifiers, 134
  - sea battle argument, 365
  - second-order logic, 319
  - self-resolving, 192
  - semantic tableaux (for p-valued logics), 334
    - (in FOL), 154
    - (PL), 54
    - (PL), algorithm, 61
    - (uses), 61
    - (with unification), 166
  - semantic
    - trees (in FOL), 186
    - trees (in PL), 96
  - semantics of PL (of PC), 41
  - semi-decision procedure, 101, 169
    - for FOL, 183
  - sequent, 90
    - calculus, 88
  - set (constructivist definition), 12
  - set operations fuzzy sets, 343
  - set theory, 10
  - sign of a literal, 46
  - skolem function, 166
    - normal form, 176
  - skolemisation, 174, 188, 326
  - sort reduction, 150
  - sound, 84
  - soundness of the resolution method, 105, 106
  - soundness of binary resolution, 192
  - standard model, 153
  - strategy, 51, 110
  - structural induction, 152, 205



structure isomorphism, 145  
 sub-formula, 134  
 substitution, 126  
   codomain, 126  
   domain, 126  
   notation, 126  
 subsumption, 266, 267  
   (rule), 93  
 supremum, 117  
 syllogism, 32

**T**

tableau  
   (closed), 56  
   (open), 56  
 tail of a rule, 219  
 tautological, 47  
 tautology, 21  
 temporal logic, 371  
 term, 125  
   algebra, 178  
   equation, 127  
   replacement, 301  
 theorem, 77  
   (Gödel's incompleteness), 92  
 torsion group, 322  
 total order, 117, 118  
   relation, 117, 507  
 totally ordered set, 118  
   (chain), 117  
 transitivity, 117  
 translation approach, 361  
 trees, 63

truth, 139  
   and satisfaction, 139  
   functional connective, 329  
   table, 42

**U**

unfair strategy, 156  
 unification, 127, 166  
   algorithm, 128  
 unit  
   clause, 46  
   strategy, 112  
 universal quantifier, 134  
 universe of discourse, 144  
 unsatisfiable, 47  
 upper bound, 117  
 uses of semantic tableaux, 61

**V**

valid, 47  
   formula in an e-frame, 390  
 valuation, 358  
 variables, 79, 135, 325  
 variant, 192  
 Venn diagrams, 114  
 very, very long proofs, 184

**W**

weak equivalence, 175  
 well-founded relation, 497  
 wff: well-founded formula, 40