

# LOGIC

ARNAUD BODIN



## Introduction

“There are true statements that cannot be proved.” This claim may sound paradoxical, yet it follows from Gödel’s celebrated Incompleteness Theorem, proved in 1931. It reveals a fundamental divide between two notions that were long treated as one: the truth of a statement and the existence of a proof of that statement. This book traces the path that leads to this discovery.

To understand this result, we must first learn the language of logic. Mathematics rests on reasoning governed by rules that we often use without stating them, but that can be described precisely. Logic allows us to see more clearly how proofs are built and which rules they rely on.

The book is divided into three parts, corresponding to three levels of difficulty:

**True/False Logic (Level 1).** This is the most elementary language of logic. We study the connectives “not,” “and,” “or,” “if... then...,” and “if and only if,” together with the rules used to construct simple proofs. This first step allows us to make the notion of logical truth more precise.

**First-Order Logic (Level 2).** Introducing the quantifiers “for all” and “there exists” allows us to express mathematical reasoning in its full generality. We then study a language rich enough to formalize a large part of everyday mathematics.

**The Incompleteness Theorem (Level 3).** This final part gradually brings together the tools needed to prove Gödel’s theorem. Readers will discover how mathematical reasoning can be applied to itself, leading to statements that are true but cannot be proved within a given system.

The first two parts are intended for any mathematics student who wants to better understand the logic underlying proofs. They can be read alongside a mathematics course: mathematical examples give the logical concepts their meaning, while logic provides a more precise framework for the reasoning encountered there. The third part is intended for motivated readers who wish to explore modern logic further and understand the proof of Gödel’s Incompleteness Theorem.

The first two parts include exercises (without solutions).

The source files are available from:

[“GitHub: Logic”](#)

# Contents

|     |                                                |     |
|-----|------------------------------------------------|-----|
| I   | True/False Logic                               | 1   |
| 1   | True/False Logic                               | 2   |
| 2   | Proofs                                         | 16  |
| 3   | Which Connectives Are Useful?                  | 33  |
| 4   | Compactness Theorem                            | 41  |
| II  | First-Order Logic                              | 50  |
| 5   | First-Order Logic                              | 51  |
| 6   | Variables and Structures                       | 61  |
| 7   | Satisfaction                                   | 73  |
| 8   | Proofs                                         | 82  |
| 9   | Soundness Theorem                              | 94  |
| 10  | Completeness Theorem                           | 101 |
| 11  | Proof of the Completeness Theorem              | 109 |
| III | Gödel's Incompleteness Theorems                | 120 |
| 12  | An Overview of Gödel's Incompleteness Theorems | 121 |
| 13  | Peano Arithmetic                               | 127 |
| 14  | Gödel Coding                                   | 137 |
| 15  | Primitive Recursive Functions                  | 148 |
| 16  | Representable Functions                        | 159 |
| 17  | Diagonalization                                | 169 |
| 18  | Proof of the Incompleteness Theorem            | 174 |
| 19  | The Second Incompleteness Theorem              | 180 |
|     | Appendix                                       | 183 |
|     | Index                                          | 190 |

# Chapter summaries

## True/False Logic

“True/false” logic, also called propositional logic, studies which of the values “True” or “False” a formula built from the logical connectives AND, OR, NOT... can take.

## Proofs

What is a proof? It is a sequence of deductions from assumptions and a small number of elementary rules, leading to a conclusion. We will describe these rules in detail and explain how to use them in practice.

## Which Connectives Are Useful?

The connectives  $\neg$  (NOT),  $\wedge$  (AND), and  $\vee$  (OR) are sufficient to construct every formula in true/false logic.

## Compactness Theorem

The Compactness Theorem states that when there are infinitely many assumptions, it is sufficient to use finitely many of them.

## First-Order Logic

We consider a language with variables and the quantifiers “for all” and “there exists.” We enrich this language with predicates and functions.

## Variables and Structures

Our goal is to use a logical language to describe concrete mathematical situations. To do so, we introduce the notion of a structure, which connects abstract formulas with actual mathematical objects. Before deciding whether, in a given structure, a formula is true or false, it is essential to distinguish free variables from bound variables.

## Satisfaction

To assign a True/False value to a first-order logic formula, we must both define a mathematical structure and assign values to the variables. We will then be able to define logical consequence.

## Proofs

We adapt the construction of proofs to the quantifiers of first-order logic.

## Soundness Theorem

We connect “proof” and “truth” through the Soundness Theorem, which states that every provable formula is true.

## Completeness Theorem

The Completeness Theorem states that, in first-order logic, everything that is true is provable. We state this theorem, its variants, and its applications.

## Proof of the Completeness Theorem

We prove the Completeness Theorem.

## An Overview of Gödel’s Incompleteness Theorems

Before embarking on the construction of the tools needed to state Gödel’s theorems precisely and prove them, we give an overview of the Incompleteness Theorems and the long road ahead.

## Peano Arithmetic

We study the axioms that define the usual arithmetic of the natural numbers.

## Gödel Coding

We turn symbols, formulas, and even proofs into natural numbers. We will then outline the proof of the First Incompleteness Theorem.

## Primitive Recursive Functions

We define and study the functions that express the “easy” computations of arithmetic.

## **Representable Functions**

We will show that Peano arithmetic can express all primitive recursive functions.

## **Diagonalization**

We bring together all the tools developed so far to prove the crucial step toward incompleteness: the diagonal lemma.

## **Proof of the Incompleteness Theorem**

We bring together all the tools developed so far to prove Gödel's First Incompleteness Theorem.

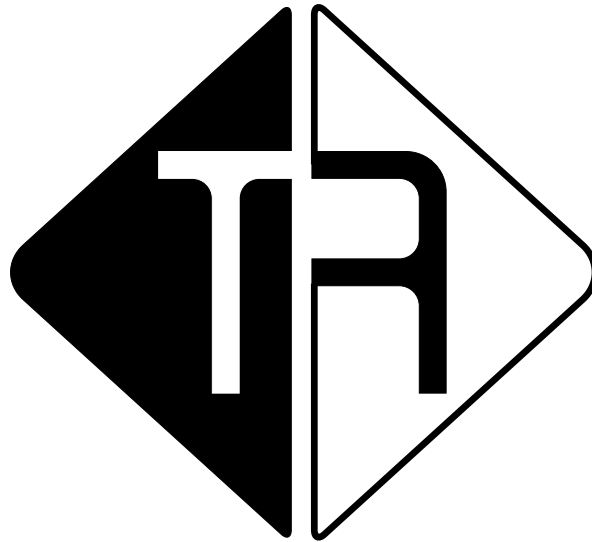
## **The Second Incompleteness Theorem**

We conclude this book with Gödel's Second Incompleteness Theorem.

---

---

# PART I



TRUE/FALSE LOGIC

---

---

# True/False Logic

“True/false” logic, also called *propositional logic*, studies which of the values “True” or “False” a formula built from the logical connectives AND, OR, NOT... can take.

## 1. True/false, connectives

### 1.1. True/false

We define two values T (True) and F (False), called **Boolean values** or **truth values**. We will also use the notation 0 (False) and 1 (True).

A **propositional variable** is a variable that can take either the value T (True) or the value F (False). For example, if  $P$  is “ $2 + 2 = 4$ ,” then  $P$  is true; in contrast,  $Q$ : “ $2^2 < 3$ ” is false.

### 1.2. $\neg$ (NOT), $\wedge$ (AND), $\vee$ (OR)

We build new formulas from one or more propositional variables and **connectives**. Let  $P$  and  $Q$  be two propositional variables.

- **Negation**  $\neg P$ .  $\neg P$  is true if and only if  $P$  is false.
- **Conjunction**  $P \wedge Q$ .  $P \wedge Q$  is true if and only if  $P$  and  $Q$  are true.
- **Disjunction**  $P \vee Q$ .  $P \vee Q$  is true if and only if  $P$  is true or  $Q$  is true.

Let us return to  $P$ : “ $2 + 2 = 4$ ” and  $Q$ : “ $2^2 < 3$ .” Then:

- $\neg P$  is false (because  $P$  is true),
- $\neg Q$  is true (because  $Q$  is false),
- $P \wedge Q$  is false (because  $Q$  is not true),
- $P \vee Q$  is true (because  $P$  is true).

Of course, there is a logical problem with our definitions: for example, we defined the connective “ $\wedge$ ” (AND) using the English word “and.” To remedy this problem, we define how each of these connectives acts through its **truth table**:

| $P$ | $\neg P$ | $P$ | $Q$ | $P \wedge Q$ | $P$ | $Q$ | $P \vee Q$ |
|-----|----------|-----|-----|--------------|-----|-----|------------|
| T   | F        | T   | T   | T            | T   | T   | T          |
| T   | F        | T   | F   | F            | T   | F   | T          |
| F   | T        | F   | T   | F            | F   | T   | T          |
|     |          | F   | F   | F            | F   | F   | F          |

Of course, we can use these connectives to build more complicated formulas:

$$\neg(P \vee Q) \quad , \quad P \wedge (\neg P) \quad , \quad P \wedge (Q \vee R) \quad , \quad (P \wedge Q) \vee R \quad , \quad \dots$$

Parentheses matter.

### 1.3. $\rightarrow$ (implication)

The *implication*  $P \rightarrow Q$  is defined by:

$$P \rightarrow Q = (\neg P) \vee Q$$

Since the connective  $\neg$  takes precedence over the others, we simply write  $\neg P \vee Q$ .

We read this as “if  $P$ , then  $Q$ ” or “ $P$  implies  $Q$ .” Its truth table is:

| $P$ | $Q$ | $P \rightarrow Q$ |
|-----|-----|-------------------|
| T   | T   | T                 |
| T   | F   | F                 |
| F   | T   | T                 |
| F   | F   | T                 |

The traditional mathematical notation for this connective is  $P \implies Q$ . But over time it has been misused to mean “therefore,” which is not what it means! We will return to this in Section 3.

Thus “ $P \rightarrow Q$ ” means “if  $P$  is true, then  $Q$  is true.” Here are a few examples:

- “it is raining”  $\rightarrow$  “the ground is wet” is a true implication. In this book, we will avoid examples in ordinary language to limit ambiguous wording and endless discussions such as “why would my kitchen floor be wet if it is raining outside?”
- “ $2+2=4$ ”  $\rightarrow$  “ $3 \times 3 = 7$ ” is a false implication.
- Let  $n$  be an integer. “ $n$  is even”  $\rightarrow$  “ $n^2$  is even” is true.

Let us turn to the surprising aspects of implication.

- “ $2+2=4$ ”  $\rightarrow$  “an integer is even or odd” is a true implication (because if  $P$  and  $Q$  are true, then  $P \rightarrow Q$  is true), even though there is no causal link between  $P$  and  $Q$ .
- “ $3 \times 3 = 7$ ”  $\rightarrow$  “ $2+2=4$ ” and “ $3 \times 3 = 7$ ”  $\rightarrow$  “ $2+2=5$ ” are also true implications. Indeed, whenever  $P$  is false,  $P \rightarrow Q$  is true. This is of course clear from the definition of implication using  $\neg P \vee Q$ . We have to accept this consequence of the definition. We will also return to it in Section 3.

### 1.4. $\leftrightarrow$ (equivalence)

The *equivalence*  $P \leftrightarrow Q$  is defined by the two implications  $P \rightarrow Q$  and  $Q \rightarrow P$ :

$$P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$$

We say “ $P$  is equivalent to  $Q$ ” or “ $P$  if and only if  $Q$ .”

In other words,  $P \leftrightarrow Q$  is true when  $P$  and  $Q$  are both true or both false.

| $P$ | $Q$ | $P \leftrightarrow Q$ |
|-----|-----|-----------------------|
| T   | T   | T                     |
| T   | F   | F                     |
| F   | T   | F                     |
| F   | F   | T                     |

Here are examples of true equivalences:

- $\neg\neg P \leftrightarrow P$ : negating a negation gives us back the original proposition.
- $P \rightarrow Q \leftrightarrow (\neg P) \vee Q$ : by the definition of implication.

In contrast, the following equivalence is *false*:

$$P \wedge Q \leftrightarrow P \vee Q$$

More precisely, this equivalence is not true for every choice of values assigned to  $P$  and  $Q$ . Indeed, the implication  $P \wedge Q \rightarrow P \vee Q$  is always true, but the implication  $P \vee Q \rightarrow P \wedge Q$  is false, for example, when  $P$  is true and  $Q$  is false.

## 1.5. Summary

There are two Boolean values T/F (True/False, or 1/0). We have defined the five basic logical connectives.

| Symbol            | Name        | Definition                                    |
|-------------------|-------------|-----------------------------------------------|
| $\neg$            | NOT         | $\neg P$ is true iff $P$ is false             |
| $\wedge$          | AND         | $P \wedge Q$ is true iff $P$ and $Q$ are true |
| $\vee$            | OR          | $P \vee Q$ is true iff $P$ or $Q$ is true     |
| $\rightarrow$     | implication | $\neg P \vee Q$                               |
| $\leftrightarrow$ | equivalence | $(P \rightarrow Q) \wedge (Q \rightarrow P)$  |

The connective  $\neg$  is a **unary** connective, because it applies to a single parameter (like a function of one variable), whereas the others are **binary** connectives (for example, we could write  $P \wedge Q$  as  $\wedge(P, Q)$ , like a function of two variables).

## 2. Formulas

### 2.1. Goals

We have **propositional variables**  $P, Q, R, \dots$ , each of which can take the value True or False. Using the connectives, we build a *formula*  $\mathcal{P}$ , for example:

$$\mathcal{P}: P \wedge (Q \vee (\neg R))$$

Here are the questions we ask. If we replace  $P, Q, R, \dots$  with values T/F, is the formula  $\mathcal{P}$  true or false? We can be even more precise:

- Is there a choice of values T/F that makes  $\mathcal{P}$  evaluate to True?
- Does  $\mathcal{P}$  evaluate to True for every possible choice of values T/F?

For example, for the proposition  $\mathcal{P}: P \wedge (Q \vee (\neg R))$ , if we assign the value T to  $P$ , F to  $Q$ , and F to  $R$ , then  $\mathcal{P}$  evaluates to  $T \wedge (F \vee (\neg F))$ , which equals T. In contrast, if we replace  $(P, Q, R)$  with (T, F, T), then  $\mathcal{P}$  evaluates to F.

Of course, we could ask the same questions about when  $\mathcal{P}$  evaluates to False, but that amounts to deciding when the formula  $\neg \mathcal{P}$  evaluates to True.

Given two formulas  $\mathcal{P}$  and  $\mathcal{Q}$ , we can build more complex formulas  $\mathcal{P} \wedge \mathcal{Q}$ ,  $\mathcal{P} \vee \mathcal{Q}$ , etc.

Given two formulas  $\mathcal{P}$  and  $\mathcal{Q}$ , our most important goal will be to understand the relationship between  $\mathcal{P}$  and  $\mathcal{Q}$ . For example, is  $\mathcal{P} \rightarrow \mathcal{Q}$  true? Is  $\mathcal{P} \leftrightarrow \mathcal{Q}$  true?

## 2.2. Definition of a formula

Consider an **alphabet** consisting of:

- a set  $\text{Var} = \{P, Q, R, \dots\}$  of propositional variables,
- a set  $\text{Conn} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$  of logical connectives,
- the parenthesis symbols ( and ).

A **formula** is a finite word made up of characters from this alphabet; it must be correctly parenthesized and use the connectives consistently.

### Definition.

More precisely, the set  $\text{Form}$  of all **formulas** is the smallest set of words containing the propositional variables such that:

- if  $\mathcal{P}$  is a formula, then so is  $\neg\mathcal{P}$ ,
- if  $\mathcal{P}$  and  $\mathcal{Q}$  are formulas, then  $(\mathcal{P} \wedge \mathcal{Q})$ ,  $(\mathcal{P} \vee \mathcal{Q})$ ,  $(\mathcal{P} \rightarrow \mathcal{Q})$ ,  $(\mathcal{P} \leftrightarrow \mathcal{Q})$  are also formulas.

Remark: we will use  $P, Q, R, \dots$  (plain letters) for propositional variables and  $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \dots$  (calligraphic letters) for formulas.

Here are some examples of formulas:

$$P \quad , \quad \neg P \quad , \quad (Q \rightarrow P) \quad , \quad ((Q \rightarrow P) \leftrightarrow Q) \quad , \quad (\neg R \rightarrow (P \vee Q))$$

With some abuse of notation:

- We will allow ourselves to omit the outermost parentheses, for example writing  $Q \rightarrow P$  instead of  $(Q \rightarrow P)$ , or to omit parentheses when there is no ambiguity, as in  $P \vee Q \vee R$ .
- Conversely, we will sometimes add unnecessary parentheses. For example, we may write  $(\neg P) \wedge Q$  instead of  $\neg P \wedge Q$  to avoid confusion with  $\neg(P \wedge Q)$ .

## 2.3. Truth tables, evaluation

For a formula  $\mathcal{P}$  that depends on two propositional variables  $P$  and  $Q$ , we can determine when the formula takes the value True or False depending on the values assigned to  $P$  and  $Q$ . We summarize this in a **truth table** for the formula  $\mathcal{P}$ :

| $P$ | $Q$ | $\mathcal{P}$ |
|-----|-----|---------------|
| T   | T   | T/F ?         |
| T   | F   | T/F ?         |
| F   | T   | T/F ?         |
| F   | F   | T/F ?         |

More generally, let  $\mathcal{P}$  be a formula in the variables  $P_1, P_2, \dots, P_n$ . An **assignment of truth values** is a choice of T/F for the variable  $P_1$ , a choice of T/F for the variable  $P_2, \dots$ . This gives an **evaluation** of the formula  $\mathcal{P}$ , with value T/F.

In practice, we construct the truth table of the formula  $\mathcal{P}$ .

### Example.

Let:

$$\mathcal{P} : \quad (P \wedge Q) \vee (\neg P)$$

We construct the truth table of  $\mathcal{P}$  using the intermediate tables for  $P \wedge Q$  and  $\neg P$ .

| $P$ | $Q$ | $P \wedge Q$ | $\neg P$ | $\mathcal{P}$ |
|-----|-----|--------------|----------|---------------|
| T   | T   | T            | F        | T             |
| T   | F   | F            | F        | F             |
| F   | T   | F            | T        | T             |
| F   | F   | F            | T        | T             |

In the chapters “Which Connectives Are Useful?” and “Compactness Theorem,” we will present an assignment of truth values as a function  $v : \{P_1, \dots, P_n\} \rightarrow \{T, F\}$  that extends to the set of formulas to give an evaluation  $v(\mathcal{P})$ .

## 2.4. Logical equivalence

### Definition.

Two formulas  $\mathcal{P}$  and  $\mathcal{Q}$  are **logically equivalent** if they have the same truth table. We write:

$$\mathcal{P} \equiv \mathcal{Q}$$

This means that the formulas  $\mathcal{P}$  and  $\mathcal{Q}$  evaluate to True at the same time and to False at the same time, whatever the choice of truth values.

There is a direct link with the equivalence connective  $\leftrightarrow$ . Indeed,  $\mathcal{P} \equiv \mathcal{Q}$  means that  $(\mathcal{P} \leftrightarrow \mathcal{Q})$  is a tautology (that is, it always evaluates to True); see Section 4.

### Example.

1.  $\neg\neg P \equiv P$  (double negation)
2.  $\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q)$  (De Morgan’s law)
3.  $(P \wedge Q) \rightarrow R \equiv (\neg(P \wedge Q)) \vee R \equiv ((\neg P) \vee (\neg Q)) \vee R$

## 2.5. Common logical equivalences

|                  |                                                                                                                                   |                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| idempotence      | $\mathcal{P} \wedge \mathcal{P} \equiv \mathcal{P}$                                                                               | $\mathcal{P} \vee \mathcal{P} \equiv \mathcal{P}$                                                                               |
| commutativity    | $\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P}$                                                            | $\mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P}$                                                              |
| associativity    | $(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R})$                  | $(\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R})$                        |
| distributivity   | $\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$ | $\mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R})$ |
| absorption       | $\mathcal{P} \wedge (\mathcal{P} \vee \mathcal{Q}) \equiv \mathcal{P}$                                                            | $\mathcal{P} \vee (\mathcal{P} \wedge \mathcal{Q}) \equiv \mathcal{P}$                                                          |
| De Morgan’s laws | $\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv (\neg \mathcal{P}) \vee (\neg \mathcal{Q})$                                          | $\neg(\mathcal{P} \vee \mathcal{Q}) \equiv (\neg \mathcal{P}) \wedge (\neg \mathcal{Q})$                                        |
| contrapositive   | $\mathcal{P} \rightarrow \mathcal{Q} \equiv (\neg \mathcal{Q}) \rightarrow (\neg \mathcal{P})$                                    |                                                                                                                                 |

These are the most familiar ones. There are many others!

### 3. Connectives (continued)

We offer some clarifications and comments on our favorite connectives.

#### 3.1. True/False, negation

There are exactly two Boolean values  $\{T, F\}$  (or  $\{0, 1\}$ ), no more and no fewer! So if a variable  $P$  or a formula  $\mathcal{P}$  does not evaluate to False, it must be True. And conversely. There is no “maybe” answer. This has the following consequences:

- *Law of excluded middle:*

$$\mathcal{P} \vee \neg \mathcal{P} \quad \text{is always True}$$

The expression “excluded middle” means that there is no choice other than T/F.

- Principle of noncontradiction:

$$\mathcal{P} \wedge \neg \mathcal{P} \quad \text{is always False}$$

- Law of double negation:

$$\neg \neg \mathcal{P} \equiv \mathcal{P}$$

A little later, in Section 4, we will see a new way of writing what is true/false:

$\top$  : a formula that always evaluates to True

This symbol denotes a formula that always evaluates to T. It is read as “always True,” “tautology,” or *top*. Its counterpart is the upside-down symbol:

$\perp$  : a formula that always evaluates to False

We can therefore write:

$$\mathcal{P} \vee \neg \mathcal{P} \equiv \top \quad \mathcal{P} \wedge \neg \mathcal{P} \equiv \perp$$

What is the difference between  $\top$  and  $T$ ?  $T$  is a value, whereas  $\top$  is a formula that always evaluates to  $T$ . If we instead identify the Boolean values with  $0/1$ , then  $1$  is a constant, whereas  $\top$  is the constant function equal to  $1$ .

#### 3.2. $\wedge$ (AND), $\vee$ (OR)

In our list of logical equivalences, we had:

$$\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P} \quad \mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P} \quad \text{commutativity}$$

and also:

$$(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R}) \quad (\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R}) \quad \text{associativity}$$

We therefore allow ourselves to write  $\mathcal{P} \wedge \mathcal{Q} \wedge \mathcal{R}$  and  $\mathcal{P} \vee \mathcal{Q} \vee \mathcal{R}$  without parentheses. Finally:

$$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R}) \quad \mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R}) \quad \text{distributivity}$$

The connective  $\wedge$  (AND) is analogous to multiplication  $\times$ , while  $\vee$  (OR) is analogous to addition  $+$ :  $a \times b = b \times a$  (commutativity),  $(a \times b) \times c = a \times (b \times c)$  (associativity),  $a \times (b + c) = a \times b + a \times c$  (distributivity).

De Morgan’s laws can be summed up by the incantations “the negation of an AND is an OR” and “the negation of an OR is an AND”:

$$\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv \neg \mathcal{P} \vee \neg \mathcal{Q} \quad \neg(\mathcal{P} \vee \mathcal{Q}) \equiv \neg \mathcal{P} \wedge \neg \mathcal{Q}$$

### 3.3. $\rightarrow$ (implication)

By definition:

$$P \rightarrow Q \equiv \neg P \vee Q$$

and its truth table is:

| $P$ | $Q$ | $P \rightarrow Q$ |
|-----|-----|-------------------|
| T   | T   | T                 |
| T   | F   | F                 |
| F   | T   | T                 |
| F   | F   | T                 |

There are several points to keep in mind about implications.

#### No causality.

It is essential to understand that  $P \rightarrow Q$  (here with propositional variables) does not indicate a causal link. Instead, it makes an observation: if  $P$  is true, is  $Q$  also true?

Here is an example that should make all your math teachers scream.

#### Example.

Let  $n$  be an integer:

$$n \text{ is odd} \rightarrow n \text{ is prime}$$

All your teachers will tell you that this implication is false, but that is not true! The confusion comes from our habit of understanding this sentence to mean “every odd number is also a prime number,” which is of course false. But here  $n$  is fixed and its value has not been specified, so the implication may be true or false depending on the value of  $n$ .

- if  $n = 3$ , then “ $n$  is odd  $\rightarrow n$  is prime” is true, because  $P$ : “3 is odd” and  $Q$ : “3 is prime” are true.
- if  $n = 15$ , then “ $n$  is odd  $\rightarrow n$  is prime” is false, because  $P$ : “15 is odd” is true whereas  $Q$ : “15 is prime” is false.
- if  $n = 4$ , then “ $n$  is odd  $\rightarrow n$  is prime” is true! Indeed,  $P$ : “4 is odd” is false, so  $P \rightarrow Q$  is true.

#### When $P$ is false, $P \rightarrow Q$ is true!

This is of course clear from the definition  $(\neg P) \vee Q$ . Let us try to explain why this is natural. Let us ask ourselves: “When do we want  $P$  implies  $Q$  to be false?” The answer is: “When  $P$  is true and  $Q$  is false” (it is like finding a counterexample). In every other case,  $P \rightarrow Q$  is true, including when  $P$  is false.

We also find this in everyday language:

“If you’re right, then I’m the pope.”

Meaning, “You’re wrong.” In the same vein:

“I’ll do it when pigs fly.”

Meaning, “I won’t do it.” The implication “pigs flying  $\rightarrow$  action” is always true, with or without the action.

#### About proofs.

From  $P \rightarrow Q$ , we cannot deduce that  $Q$  is true. What we will see next is called *modus ponens*, and it is the basis of our reasoning:

If  $P$  is true and  $P \rightarrow Q$  is true, then  $Q$  is true.

This is the mistake we make out of habit when we write  $P \implies Q$ : we implicitly assume that  $P$  is true and deduce that  $Q$  is true.

Finally,  $\mathcal{P} \rightarrow \mathcal{Q} \rightarrow \mathcal{R}$  does not denote a chain of consequences; by definition, it is the formula:

$$\mathcal{P} \rightarrow (\mathcal{Q} \rightarrow \mathcal{R})$$

### 3.4. $\leftrightarrow$ (equivalence)

For an equivalence to be true,  $\mathcal{P}$  and  $\mathcal{Q}$  must be true at the same time and false at the same time. By definition:

$$\mathcal{P} \leftrightarrow \mathcal{Q} \equiv (\mathcal{P} \rightarrow \mathcal{Q}) \wedge (\mathcal{Q} \rightarrow \mathcal{P})$$

We can give the same warnings as for implication: writing  $\mathcal{P} \leftrightarrow \mathcal{Q}$  does not mean that  $\mathcal{P}$  and  $\mathcal{Q}$  always evaluate to True, only that they evaluate to True at the same time.

### 3.5. Other connectives

An  *$n$ -ary connective* is a function  $f : \{T, F\}^n \rightarrow \{T, F\}$ , or, if you prefer, a function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ .

- $n = 1$ .  $\neg$  is a unary connective (a function of one variable). We may also need the identity  $\text{id}$ , which maps T to T and F to F.
- $n = 2$ .  $\wedge$  is a binary connective (like a function of two variables). Except that we write  $P \wedge Q$  instead of  $\wedge(P, Q)$ .
- $n = 2$ . **Exclusive OR**  $\oplus/\text{XOR}$  is another binary connective:  
 $P \oplus Q$  is true if exactly one of the variables is true.
- $n = 2$ . The connective  $\uparrow/\text{NAND}$ , also called the Sheffer stroke, is defined as the negation of AND:

$$P \uparrow Q = \neg(P \wedge Q)$$

We will list all binary connectives in the chapter “Which Connectives Are Useful?”

- For any  $n$ , we can define an  $n$ -ary connective by giving its truth table in terms of the values assigned to the variables  $P_1, P_2, \dots, P_n$ . This table has  $2^n$  rows. We will see why we do not really need these connectives in that same chapter, “Which Connectives Are Useful?”

## 4. Formulas (continued)

### 4.1. Satisfiable formulas

Let  $P_1, P_2, \dots, P_n$  be propositional variables. An assignment of truth values  $v$  associates with each variable  $P_i$  a truth value  $v(P_i)$ , which is T or F (or a value 0/1). We can then associate with any formula  $\mathcal{P}$  a value T/F, denoted by  $v(\mathcal{P})$ .

#### Example.

Consider three variables  $P, Q, R$  and the formula:

$$\mathcal{P} : (P \wedge \neg Q) \vee (Q \wedge R)$$

Consider the assignment of truth values (T, F, F) (that is,  $v(P) = T$ ,  $v(Q) = F$ ,  $v(R) = F$ ). Substituting these values into  $\mathcal{P}$  gives:

$$v(\mathcal{P}) = (T \wedge \neg F) \vee (F \wedge F)$$

so  $v(\mathcal{P}) = T$ . In contrast, for the assignment (T, T, F),  $\mathcal{P}$  evaluates to F.

**Definition.**

- A formula  $\mathcal{P}$  is **satisfied** by an assignment of truth values  $v$  if  $\mathcal{P}$  evaluates to True for that assignment.
- A formula  $\mathcal{P}$  is **satisfiable** if there is an assignment of truth values for which  $\mathcal{P}$  evaluates to True.

The formula  $\mathcal{P} : (P \wedge \neg Q) \vee (Q \wedge R)$  in the previous example is satisfiable, because we found an assignment of truth values that makes  $\mathcal{P}$  true.

**4.2. Tautology****Definition.**

- A **tautology** is a formula  $\mathcal{P}$  that always evaluates to True, regardless of the assignment of truth values.
- A **contradiction** is a formula  $\mathcal{P}$  that always evaluates to False, regardless of the assignment of truth values.

Notation:

|                    |                         |
|--------------------|-------------------------|
| tautology : $\top$ | contradiction : $\perp$ |
|--------------------|-------------------------|

A few remarks:

- Thus a tautology is a formula that is always true (more precisely, one that always evaluates to the value True). The term comes from the Greek *tautos* (the same) and *logos* (speech): it is always the same value (True).
- A contradiction is a formula that is always false.
- However, most formulas are neither tautologies nor contradictions: they evaluate to true or false values depending on the assignment of truth values.
- A formula  $\mathcal{P}$  is a contradiction if and only if  $\neg \mathcal{P}$  is a tautology. This is why we will focus mainly on tautologies.

**Example.**

1.  $P \vee \neg P$  is a tautology (if  $P$  is True,  $P \vee \neg P$  is True; if  $P$  is False,  $P \vee \neg P$  is also True).
2.  $P \wedge \neg P$  is a contradiction (it is a formula that is always false, whatever the value assigned to  $P$ ).
3.  $(P \vee Q \vee R) \vee (\neg P \wedge \neg Q \wedge \neg R)$  is a tautology.
4.  $(P \wedge Q) \rightarrow Q$  is a tautology.
5.  $\neg(P \wedge Q) \leftrightarrow (\neg P) \vee (\neg Q)$  is a tautology.
6.  $((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R)$  is a tautology.

Here are examples of how the symbols  $\top/\perp$  are used:

$$\mathcal{P} \wedge \top \equiv \mathcal{P} \quad \mathcal{P} \vee \top \equiv \top \quad \mathcal{P} \wedge \perp \equiv \perp \quad \mathcal{P} \vee \perp \equiv \mathcal{P}$$

We can now make the link between logical equivalence ( $\equiv$ ) and equivalence ( $\leftrightarrow$ ) precise:

$$\mathcal{P} \equiv \mathcal{Q} \quad \text{iff} \quad \mathcal{P} \leftrightarrow \mathcal{Q} \text{ is a tautology}$$

If this is the case, that is, if  $P \equiv Q$  (or if  $P \leftrightarrow Q$  is always true), then  $P$  and  $Q$  evaluate to True at the same time and to False at the same time, whatever the evaluation.

In the following sections, we will discover three other ways of viewing formulas and their evaluations.

### 4.3. Logic gates and circuits

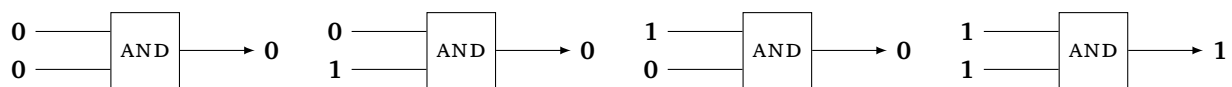
To get closer to what happens inside a computer, we can think of connectives as logic gates. Here the truth values are  $\{0, 1\}$ .

**NOT gate.**

It has a single input and changes 0 to 1 and 1 to 0.

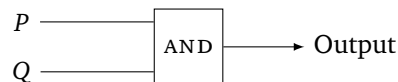


**AND gate.**

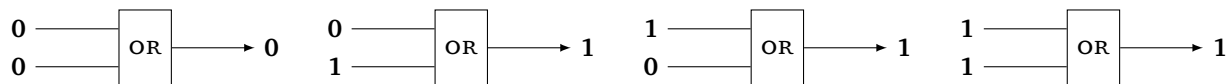


The following table summarizes how the AND gate acts:

| $P$ | $Q$ | Output |
|-----|-----|--------|
| 0   | 0   | 0      |
| 0   | 1   | 0      |
| 1   | 0   | 0      |
| 1   | 1   | 1      |



**OR gate.**



**NAND gate.**

It can be implemented as the gate  $\text{NOT}(\text{AND})$ , that is, an AND gate followed by a NOT gate.

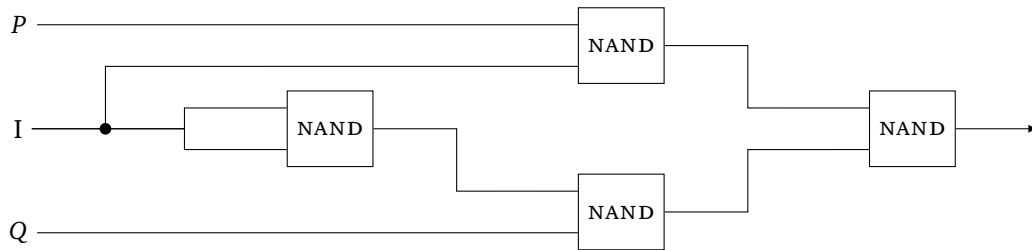


But we prefer to see it as a gate in its own right.



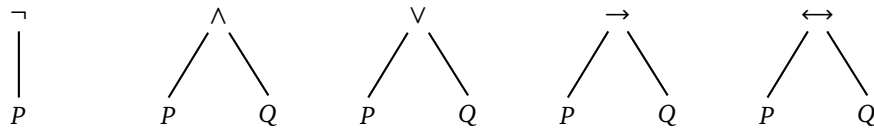
#### Example.

A *multiplexer* is a circuit that uses a selector  $I$  to select a signal from one of two inputs  $P$  and  $Q$ . If  $I$  is 1, the circuit's output is the value of  $P$ . If  $I$  is 0, the output is the value of  $Q$ . We can also view this as a rudimentary test: "If  $I = 1$ , do this; otherwise, do that."



## 4.4. Trees

We can also represent connectives as *trees*:

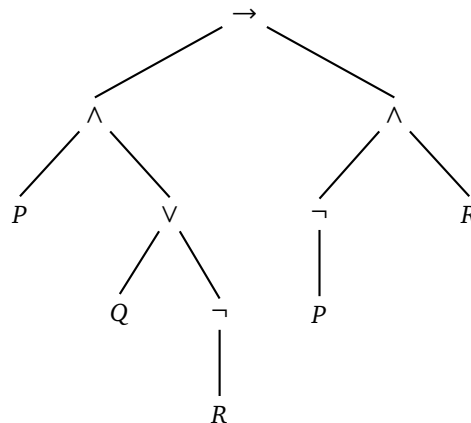


### Example.

Consider the formula:

$$(P \wedge (Q \vee \neg R)) \rightarrow (\neg P \wedge R)$$

The corresponding tree is:



The advantage is that a tree structure is well suited to algorithmic processing. A preorder depth-first traversal visits the root, then explores the left subtree, then the right subtree.

We can also adopt *Polish notation*, that is, write  $\wedge PQ$  (like a function of two variables  $\wedge(P, Q)$ ) instead of  $P \wedge Q$ . In Polish notation, the formula in the previous example is written:

$$\rightarrow \wedge P \vee Q \neg R \wedge \neg P R$$

This expression is particularly well suited to computer processing, because it requires no parentheses. For a human, it is easier to read when written as follows:

$$\rightarrow (\wedge (P)(\vee (Q)(\neg R)))(\wedge (\neg P)(R))$$

## 4.5. Algebraic formulas

We denote the Boolean values by  $\mathbb{B} = \{0, 1\}$ . In this section, we calculate modulo 2, that is, we write  $1 + 1 = 0$  (strictly speaking, we should write  $1 + 1 \equiv 0 \pmod{2}$ ). Note that modulo 2, we have  $+1 = -1$ . Let  $x, y \in \{0, 1\}$ . Here are the algebraic expressions that reproduce the action of the connectives:

$$\begin{aligned}\neg x &= 1 + x \\ x \wedge y &= x \times y \\ x \vee y &= x + y - xy && \text{inclusive OR} \\ x \oplus y &= x + y && \text{exclusive OR}\end{aligned}$$

This is easy to justify. For example, if  $x = 0$ , then  $1 + x = 1$ , and if  $x = 1$ , then  $1 + x = 1 + 1 = 0$ . So  $1 + x$  does indeed correspond to  $\neg x$ .

Similarly,  $x \times y$  can equal 1 only if  $x = 1$  and  $y = 1$ .

### Example.

These algebraic expressions can provide proofs. For example, double negation is justified by:

$$\neg\neg x = \neg(1 + x) = 1 + (1 + x) = 1 + 1 + x = x$$

Here is a proof of De Morgan's law for the negation of an AND,  $\neg(x \wedge y) = (\neg x) \vee (\neg y)$ :

$$(\neg x) \vee (\neg y) = (1 + x) \vee (1 + y) = (1 + x) + (1 + y) - (1 + x)(1 + y) = x + y - 1 - x - y - xy = 1 + xy = \neg(xy)$$

We used  $-1 = +1$ , so  $-xy = xy$ .

## Exercises

### True/false, connectives

**Exercise 1.** Let  $P$  be " $2 \times 2 = 5$ " and  $Q$  be " $4 + 4 > 7$ ." State whether the following propositions are true or false:  $\neg P$ ,  $\neg Q$ ,  $P \wedge Q$ ,  $P \vee Q$ ,  $P \rightarrow Q$ ,  $Q \rightarrow P$ ,  $P \leftrightarrow Q$ .

**Exercise 2.** Assume that  $P$  is true. For which values of  $Q$  are the following propositions true:  $P \wedge Q$ ,  $P \vee Q$ ,  $P \rightarrow Q$ ,  $Q \rightarrow P$ ,  $P \leftrightarrow Q$ ? Answer the same question assuming this time that  $P$  is false.

### Formulas

**Exercise 3.** Which of the following words are formulas (in the strict sense of the definition)?

$$\neg\neg P \quad (\neg P \vee Q \wedge R) \quad (P \rightarrow (Q \wedge (R \wedge S))) \quad (P, Q \rightarrow R) \quad ((P \wedge \neg Q \rightarrow)(Q \vee R))$$

**Exercise 4.** Construct the truth table for  $\neg(P \vee Q)$  and the one for  $(\neg P) \wedge (\neg Q)$ . Deduce De Morgan's law for  $\vee$ .

**Exercise 5.** Prove the following associativity and distributivity formulas:

$$P \vee (Q \vee R) \equiv (P \vee Q) \vee R \quad P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

**Exercise 6.** Show the following logical equivalences:

$$P \rightarrow Q \wedge R \equiv (P \rightarrow Q) \wedge (P \rightarrow R) \quad P \vee Q \rightarrow R \equiv (P \rightarrow R) \wedge (Q \rightarrow R) \quad P \leftrightarrow Q \equiv (P \vee Q) \rightarrow (P \wedge Q)$$

Express these equivalences in a sentence in English.

## Connectives (continued)

**Exercise 7.** Show that the following formulas are tautologies (they are always true, whatever the values of  $P$  and  $Q$ ):

$$P \vee (P \rightarrow Q) \quad (P \wedge Q) \rightarrow P \quad (P \rightarrow Q) \vee (P \rightarrow \neg Q) \quad (P \rightarrow Q) \vee (\neg P \rightarrow Q)$$

**Exercise 8.** For each of the following formulas, find a logically equivalent formula whose expression is as simple as possible:

$$Q \vee (P \wedge Q) \quad P \wedge (P \vee Q) \quad P \wedge (\neg P \vee Q) \quad (P \vee Q) \rightarrow (P \wedge Q)$$

**Exercise 9.** Let  $x \in \mathbb{R}$ , let  $P$  be “ $x \geq 2$ ,” and let  $Q$  be “ $x^2 \geq 4$ .” Find all values  $x \in \mathbb{R}$  for which  $P \rightarrow Q$  is true. Answer the same question for  $Q \rightarrow P$ , then for  $P \leftrightarrow Q$ .

**Exercise 10.**

1. Determine the truth table of  $P \rightarrow (Q \rightarrow R)$ , then that of  $(P \rightarrow Q) \rightarrow R$ . What do you conclude?
2. Show that  $P \leftrightarrow (Q \leftrightarrow R) \equiv (P \leftrightarrow Q) \leftrightarrow R$ .
3. Show that  $P \rightarrow (Q \rightarrow R) \equiv (P \wedge Q) \rightarrow R \equiv Q \rightarrow (P \rightarrow R)$ .

**Exercise 11.** Recall that  $P \oplus Q$  denotes “exclusive OR” (it is true when  $P$  is true or  $Q$  is true, but not both).

1. Calculate the truth table of  $P \oplus Q$ .
2. Show that  $P \oplus Q$  amounts to “ $P \neq Q$ ,” that is,  $P \oplus Q \equiv (P \wedge \neg Q) \vee (\neg P \wedge Q)$ . What is the relationship between  $\oplus$  and  $\leftrightarrow$ ?
3. Show that  $\oplus$  is commutative and associative.
4. Express  $P \wedge (Q \oplus R)$  using only the connectives  $\wedge$ ,  $\vee$ , and  $\neg$ . Do the same for  $P \oplus (Q \wedge R)$ .

**Exercise 12.** Consider  $P_1, P_2, \dots, P_n$ , with  $n \geq 1$ . When is  $P_1 \oplus P_2 \oplus \dots \oplus P_n$  true?

## Formulas (continued)

**Exercise 13.** Are the following formulas satisfiable? Are they tautologies?

$$P \wedge Q \rightarrow P \quad (P \vee Q) \wedge (\neg P \vee \neg Q) \quad P \vee (Q \wedge \neg Q) \leftrightarrow P \quad (P \rightarrow Q) \wedge (Q \rightarrow \neg P) \\ ((P \rightarrow Q) \wedge (R \rightarrow S)) \rightarrow ((P \vee R) \rightarrow (Q \vee S))$$

**Exercise 14.** Show that the following formulas are tautologies:

1.  $P \wedge (P \rightarrow Q) \rightarrow Q$  (*modus ponens*)
2.  $(P \rightarrow Q) \wedge \neg Q \rightarrow \neg P$  (*modus tollens*)
3.  $(P \rightarrow Q) \leftrightarrow (\neg Q \rightarrow \neg P)$  (*contraposition*)
4.  $(P \rightarrow Q) \leftrightarrow ((P \wedge \neg Q) \rightarrow \perp)$  (*proof by contradiction*)
5.  $((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R)$  (*sylogism/transitivity of implication*).
6.  $(P \wedge \top) \leftrightarrow P$

**Exercise 15.**

1. Construct the circuit for  $P \wedge (Q \vee R)$  and the one for  $(P \wedge Q) \vee (P \wedge R)$ . Check that they produce the same outputs.
2. Show that using only NAND gates, we can build circuits whose outputs are identical to the action of  $\neg$ ,  $\wedge$ , and  $\vee$ .

**Exercise 16.** For each of the following formulas, construct the corresponding tree and give its expression in Polish notation:

$$\neg P \wedge (Q \vee (P \wedge R)) \qquad ((P \vee Q) \wedge \neg R) \rightarrow (P \wedge (\neg Q \wedge R))$$

**Exercise 17.**

1. Justify the algebraic formula  $x \vee y = x + y - xy$ .
2. Using the algebraic formulas, show:

$$x \wedge (y \wedge z) = (x \wedge y) \wedge z \qquad x \vee (y \vee z) = (x \vee y) \vee z \qquad x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

3. Calculate  $x \wedge (y \oplus z)$  and  $(x \wedge y) \oplus (x \wedge z)$ . What do you conclude?
4. Find an algebraic formula for  $x \rightarrow y$ . Do the same for  $x \leftrightarrow y$ .
5. Using the algebraic formulas, show that  $x \rightarrow y = \neg y \rightarrow \neg x$ .

*What is a proof? It is a sequence of deductions from assumptions and a small number of elementary rules, leading to a conclusion. We will describe these rules in detail and explain how to use them in practice.*

## 1. Logical consequence

### 1.1. Assumptions & conclusion

We have formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$ , which will be our **assumptions** (or **premises**), and a formula  $\mathcal{Q}$ , which will be our **conclusion**. Our aim is, of course, to move from true assumptions to a true conclusion. There will be two ways of doing this, and this is truly at the heart of Gödel's theory.

*Remark.* If there are finitely many assumptions  $\mathcal{P}_1, \dots, \mathcal{P}_l$ , we could always reduce them to the single assumption  $\mathcal{P}_1 \wedge \mathcal{P}_2 \wedge \dots \wedge \mathcal{P}_l$ ; in practice, we will not do this. We might also want several formulas as conclusions, but we will not do that either, since it would amount to studying them one by one.

### 1.2. Logical consequence $\models$

Let us begin with the notion of *logical consequence*, also called *semantic consequence*.

#### Definition.

We say that  $\mathcal{Q}$  is a **logical consequence** of  $\mathcal{P}_1, \dots, \mathcal{P}_l$  if every valuation that makes the formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  true also makes the formula  $\mathcal{Q}$  true. We write:

$$\boxed{\mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q}}$$

The symbol  $\models$  is the *double turnstile*; it is read “entails.” In other words, if the assumptions hold, then the conclusion is true. (If one of the assumptions  $\mathcal{P}_i$  is false, then nothing is required of  $\mathcal{Q}$ , which may be true or false.)

### 1.3. Truth table

Establishing a logical consequence

$$\mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q}$$

is a matter of checking. We construct the truth table for all the propositional variables  $P_1, \dots, P_n$  occurring in the formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  and  $\mathcal{Q}$ . In every row in which all the formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  evaluate to True,  $\mathcal{Q}$  must also evaluate to True.

## 1.4. Examples

### Example.

$$P \vee Q, \neg Q \models P$$

Here we have:

- two propositional variables  $P$  and  $Q$ ,
- two assumptions:  $\mathcal{P}_1$ : “ $P \vee Q$ ” and  $\mathcal{P}_2$ : “ $\neg Q$ ”,
- one conclusion: “ $P$ .”

Here is the truth table:

| $P$ | $Q$ | $P \vee Q$ | $\neg Q$ | $P$ |
|-----|-----|------------|----------|-----|
| T   | T   | T          | F        | —   |
| T   | F   | T          | T        | T   |
| F   | T   | T          | F        | —   |
| F   | F   | F          | T        | —   |

Only the second row makes both assumptions true, and we then check that the conclusion is also true. We do not need to know whether the conclusion is true or false in the other cases. Thus we do indeed have the logical consequence  $P \vee Q, \neg Q \models P$ .

Let us emphasize that when we write

$$\mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q}$$

we do not require the formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  always to evaluate to True. What we require is that *when* they evaluate to True, so does  $\mathcal{Q}$ . An extreme situation is, for example,

$$P, \neg P \models \text{anything}$$

This logical consequence is valid, whatever the conclusion, because  $P$  and  $\neg P$  never evaluate to true at the same time.

### Example.

$$P \rightarrow Q, Q \rightarrow R \models P \rightarrow R$$

Here we have three propositional variables  $P, Q, R$ , two assumptions, and one conclusion.

| $P$ | $Q$ | $R$ | $P \rightarrow Q$ | $Q \rightarrow R$ | $P \rightarrow R$ |
|-----|-----|-----|-------------------|-------------------|-------------------|
| T   | T   | T   | T                 | T                 | T                 |
| T   | T   | F   | T                 | F                 | —                 |
| T   | F   | T   | F                 | T                 | —                 |
| T   | F   | F   | F                 | T                 | —                 |
| F   | T   | T   | T                 | T                 | T                 |
| F   | T   | F   | T                 | F                 | —                 |
| F   | F   | T   | T                 | T                 | T                 |
| F   | F   | F   | T                 | T                 | T                 |

We consider only the rows in which both assumptions hold, and then check that the conclusion is also

true. We do not need to know whether the conclusion is true or false in the other cases. Thus we do indeed have the desired logical consequence.

In the following example, we do not have a logical consequence!

**Example.**

Let the assumptions be:

$$\neg P, P \vee Q$$

Do they entail

$$P \wedge Q?$$

We construct the truth table:

| $P$      | $Q$      | $\neg P$ | $P \vee Q$ | $P \wedge Q$ |
|----------|----------|----------|------------|--------------|
| $\vdots$ | $\vdots$ |          |            |              |
| F        | T        | T        | T          | F            |
| $\vdots$ | $\vdots$ |          |            |              |

We see that one row is problematic: if  $P$  is false and  $Q$  is true, then both assumptions are true, yet the desired conclusion is false. Thus  $P \wedge Q$  is not a logical consequence of the assumptions  $\neg P$  and  $P \vee Q$ .

## 1.5. Limitations

We have just seen that if there are  $n$  variables  $P_1, \dots, P_n$  in our assumptions  $\mathcal{P}_1, \dots, \mathcal{P}_l$  and conclusion  $Q$ , then we must construct a truth table with  $2^n$  rows to verify the logical consequence.

As  $n$  grows,  $2^n$  grows exponentially (since  $2^n = e^{n \ln 2}$ ), and constructing such a table by hand becomes impractical. More significantly, we are still working within the fairly limited setting of true/false logic. This truth-table method will no longer be possible once we introduce the parameters of first-order logic. Consider the following problem, where  $n \in \mathbb{N}$  is an integer. We will all agree that we have the following logical consequence:

$$n > 0, n \text{ is even} \models n^2 - 1 \text{ is odd}$$

But if we want to justify it with a truth table, the table has infinitely many rows depending on the parameter  $n$ , and this cannot be accepted as a proof.

| $n$      | $n > 0$  | $n \text{ is even}$ | $n^2 - 1 \text{ is odd}$ |
|----------|----------|---------------------|--------------------------|
| 0        | F        | T                   | —                        |
| 1        | T        | F                   | —                        |
| 2        | T        | T                   | T                        |
| 3        | T        | F                   | —                        |
| 4        | T        | T                   | T                        |
| $\vdots$ | $\vdots$ | $\vdots$            | $\vdots$                 |

## 2. Formal proof

### 2.1. What is a proof?

A proof aims to convince others that a statement is true. A proof must have a finite number of steps. Anyone must be able to check each of these steps, and the transition from one step to the next must be easy to understand and require no deep thought. There is a natural similarity between an algorithm (a sequence of elementary instructions intended to be read by a machine) and a proof (intended for a human). With the advent of proof assistants, this difference is fading.

### 2.2. Formal proof $\vdash$

Let  $\mathcal{P}_1, \dots, \mathcal{P}_l$  be formulas, which will be our assumptions, and let  $\mathcal{Q}$  be a formula (the conclusion). We write

$$\boxed{\mathcal{P}_1, \dots, \mathcal{P}_l \vdash \mathcal{Q}}$$

if there is a *proof* of  $\mathcal{Q}$  from  $\mathcal{P}_1, \dots, \mathcal{P}_l$ . This expression is then called a *theorem*. The symbol  $\vdash$  is the *turnstile* and is read “proves” or “allows us to prove.”

But how do we construct a proof? It is a finite sequence of formulas that starts from the assumptions and leads to the conclusion. Each line of the proof is obtained by applying one of a small number of permitted logical rules to earlier lines.

More precisely:

**Definition.**

A *proof* is a finite sequence of formulas  $\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_N$  in which each  $\mathcal{A}_i$  is:

- one of the assumptions  $\mathcal{P}_1, \dots, \mathcal{P}_l$  in the statement, or
- a consequence obtained by applying the *rules of inference* to earlier lines  $\mathcal{A}_1, \dots, \mathcal{A}_{i-1}$ .
- And  $\mathcal{A}_N$  is the conclusion  $\mathcal{Q}$ .

The rules of inference will be described in more detail below. For example, a rule may come from elementary logical formulas such as:

- if  $\mathcal{P} \wedge \mathcal{Q}$ , then  $\mathcal{P}$ ,
- if  $\mathcal{P}$  and  $\mathcal{Q}$ , then  $\mathcal{P} \wedge \mathcal{Q}$ ,
- ...

Let us summarize the work of a mathematician. The mathematician states a theorem:

**Theorem.**

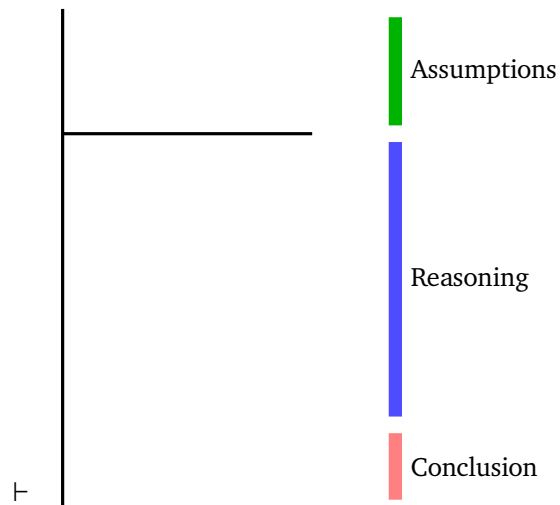
$$\mathcal{P}_1, \dots, \mathcal{P}_l \vdash \mathcal{Q}$$

Then the mathematician gives its proof.

- Remark. Here, “theorem” means a proved statement. We make no qualitative distinction between Lemma/Proposition/Theorem/Corollary according to their importance or interest.
- Warning. At first sight, no notion of true or false appears explicitly in this section on proofs. The connection with true/false logic is actually implicit: it lies in the choice of rules of inference, which were designed to respect this logic.

## 2.3. A first example

A *natural deduction proof* is presented as an array containing a sequence of lines: first the assumptions, then the reasoning, and finally the conclusion.



Before listing the rules of inference, here is a practical example of a statement and its proof.

**Theorem.**

$$\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \vdash (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$$

Proof of the theorem.

|                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                        |                                         |                                  |                                                                        |                                                                          |                                     |  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------|-----------------------------------------|----------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------|--|
| $\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R})$                                                                                                                                                                                                                                                                                                                                                                                | Assumption                                                             |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $\mathcal{P}$                                                                                                                                                                                                                                                                                                                                                                                                                      | Left-hand term of the assumption                                       |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $\mathcal{Q} \vee \mathcal{R}$                                                                                                                                                                                                                                                                                                                                                                                                     | Right-hand term of the assumption                                      |                                         |                                  |                                                                        |                                                                          |                                     |  |
| <table> <tr> <td><math>\mathcal{Q}</math></td><td>Temporary assumption. First case of OR</td></tr> <tr> <td><math>\mathcal{P} \wedge \mathcal{Q}</math></td><td>Because we already have <math>\mathcal{P}</math>, and here we have <math>\mathcal{Q}</math></td></tr> <tr> <td><math>(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})</math></td><td>Because the left-hand term is true</td></tr> </table>   | $\mathcal{Q}$                                                          | Temporary assumption. First case of OR  | $\mathcal{P} \wedge \mathcal{Q}$ | Because we already have $\mathcal{P}$ , and here we have $\mathcal{Q}$ | $(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$ | Because the left-hand term is true  |  |
| $\mathcal{Q}$                                                                                                                                                                                                                                                                                                                                                                                                                      | Temporary assumption. First case of OR                                 |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $\mathcal{P} \wedge \mathcal{Q}$                                                                                                                                                                                                                                                                                                                                                                                                   | Because we already have $\mathcal{P}$ , and here we have $\mathcal{Q}$ |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$                                                                                                                                                                                                                                                                                                                                                           | Because the left-hand term is true                                     |                                         |                                  |                                                                        |                                                                          |                                     |  |
| <table> <tr> <td><math>\mathcal{R}</math></td><td>Temporary assumption. Second case of OR</td></tr> <tr> <td><math>\mathcal{P} \wedge \mathcal{R}</math></td><td>Because we already have <math>\mathcal{P}</math>, and here we have <math>\mathcal{R}</math></td></tr> <tr> <td><math>(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})</math></td><td>Because the right-hand term is true</td></tr> </table> | $\mathcal{R}$                                                          | Temporary assumption. Second case of OR | $\mathcal{P} \wedge \mathcal{R}$ | Because we already have $\mathcal{P}$ , and here we have $\mathcal{R}$ | $(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$ | Because the right-hand term is true |  |
| $\mathcal{R}$                                                                                                                                                                                                                                                                                                                                                                                                                      | Temporary assumption. Second case of OR                                |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $\mathcal{P} \wedge \mathcal{R}$                                                                                                                                                                                                                                                                                                                                                                                                   | Because we already have $\mathcal{P}$ , and here we have $\mathcal{R}$ |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $(\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$                                                                                                                                                                                                                                                                                                                                                           | Because the right-hand term is true                                    |                                         |                                  |                                                                        |                                                                          |                                     |  |
| $\vdash (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$                                                                                                                                                                                                                                                                                                                                                    | Conclusion. Both cases lead to the same result.                        |                                         |                                  |                                                                        |                                                                          |                                     |  |

The rules we used are the introduction and elimination rules for AND and OR:

$$\mathcal{P} \wedge \mathcal{Q} \quad \text{means} \quad \mathcal{P} \text{ and } \mathcal{Q}$$

$$\mathcal{P} \vee \mathcal{Q} \quad \text{means} \quad \mathcal{P} \text{ or } \mathcal{Q}$$

### 3. Rules of inference

Here we give the list of rules permitted in a proof. Some are natural, while others are a little less obvious. We begin with a set of six *basic rules* that form the foundation of proofs in classical logic. We will then present *derived rules*, which follow from the basic rules and make proofs easier to write.

#### 3.1. The basic rules

**Rule 1. Reuse.**

$$\boxed{\mathcal{P} \vdash \mathcal{P}}$$

Reuse simply means that we may reuse a formula obtained earlier.

**Rule 2. Conjunction.**

$$\boxed{\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P} \quad \text{and} \quad \mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}}$$

$$\boxed{\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}}$$

The rules for AND are natural. For example, if we have already obtained both  $\mathcal{P}$  and  $\mathcal{Q}$ , then we have proved  $\mathcal{P} \wedge \mathcal{Q}$ .

**Rule 3. Double negation.**

$$\boxed{\neg\neg\mathcal{P} \vdash \mathcal{P}}$$

This rule will be discussed in a remark below.

Here is how these rules are written in a natural deduction proof:

$$\begin{array}{ccc} \begin{array}{|l} \vdots \\ \mathcal{P} \\ \vdots \\ \mathcal{P} \\ \vdots \end{array} & \vdash & \begin{array}{|l} \vdots \\ \mathcal{P} \wedge \mathcal{Q} \\ \vdots \\ \mathcal{P} \end{array} \\ & & \vdash \\ & & \begin{array}{|l} \vdots \\ \mathcal{P} \\ \vdots \\ \mathcal{Q} \\ \vdots \\ \mathcal{P} \wedge \mathcal{Q} \end{array} \end{array} \quad \begin{array}{ccc} \begin{array}{|l} \vdots \\ \mathcal{P} \\ \vdots \\ \neg\neg\mathcal{P} \\ \vdots \\ \mathcal{P} \end{array} & \vdash & \begin{array}{|l} \vdots \\ \mathcal{P} \end{array} \end{array}$$

When writing a proof, we may use any formula obtained earlier. We can, of course, apply several rules in succession before reaching the desired final conclusion, but each line corresponds to the application of a single rule.

**Rule 4. Disjunction.**

$$\boxed{\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q} \quad \text{and} \quad \mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}}$$

The converse direction is harder to write formally: “ $\mathcal{P} \vee \mathcal{Q}$  proves  $\mathcal{P}$  or  $\mathcal{Q}$ .” To express it correctly, we introduce an auxiliary formula  $\mathcal{R}$  as the conclusion of implications:

$$\boxed{\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{R} \quad \vdash \quad \mathcal{R}}$$

In practice, this is fairly easy (the array on the right below): if we have  $\mathcal{P} \vee \mathcal{Q}$ , we consider two cases. In the first case, we assume  $\mathcal{P}$ ; in the second, we assume  $\mathcal{Q}$ . If we prove the same conclusion  $\mathcal{R}$  in both cases, then  $\mathcal{R}$  is proved.

$$\begin{array}{c|c} \vdots & \\ \mathcal{P} & \\ \vdots & \\ \vdash \quad \mathcal{P} \vee \mathcal{Q} & \end{array} \quad \begin{array}{c|c|c} \mathcal{P} \vee \mathcal{Q} & & \\ \hline \mathcal{P} & \text{First case of OR. Temporary assumption } \mathcal{P} & \\ \hline \mathcal{R} & & \\ \hline \mathcal{Q} & \text{Second case of OR. Temporary assumption } \mathcal{Q} & \\ \hline \mathcal{R} & & \\ \hline \vdash \quad \mathcal{R} & & \end{array}$$

### Rule 5. Implication.

$$\boxed{\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \quad \vdash \quad \mathcal{Q}}$$

This rule is called *modus ponens*. It is a kind of elementary mini-proof: if we have an assumption  $\mathcal{P}$  and know that  $\mathcal{P}$  implies  $\mathcal{Q}$ , then we have proved  $\mathcal{Q}$ !

The converse operation, namely introducing the implication, is harder to express:

$$\boxed{\text{If } \mathcal{P} \vdash \mathcal{Q} \quad \text{then} \quad \vdash \mathcal{P} \rightarrow \mathcal{Q}.}$$

If we can prove  $\mathcal{Q}$  under the assumption  $\mathcal{P}$ , then we have a proof of  $\mathcal{P} \rightarrow \mathcal{Q}$ .

Fortunately, this second rule is again easy to understand and use with a proof array (the figure on the right below).

$$\begin{array}{c|c} \vdots & \\ \mathcal{P} & \\ \vdots & \\ \mathcal{P} \rightarrow \mathcal{Q} & \\ \vdots & \\ \vdash \quad \mathcal{Q} & \end{array} \quad \begin{array}{c|c|c} & \mathcal{P} & \text{Temporary assumption} \\ \hline & \mathcal{Q} & \text{Conclusion under this assumption} \\ \hline \vdash \quad \mathcal{P} \rightarrow \mathcal{Q} & & \end{array}$$

### Rule 6. Explosion.

$$\boxed{\perp \quad \vdash \quad \mathcal{P}}$$

The rule of explosion (*ex falso*) says: “False implies anything.” This may seem a little strange, but it will be useful, for example, in proofs by contradiction.

**Remark.**

- We will state other rules below, but these rules can be derived from the basic rules.
- For each of the connectives  $\wedge$ ,  $\vee$ ,  $\rightarrow$ , there are two associated rules: an *elimination* rule that removes the symbol (for example,  $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$ ) and an *introduction* rule that introduces the symbol (for example,  $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$ ).
- The basic rules just stated correspond to what is called *classical logic*. These are conventional choices that agree with our mathematical experience, but other approaches are possible.
- *Minimal logic* rejects the rules  $\neg\neg\mathcal{P} \vdash \mathcal{P}$  and  $\perp \vdash \mathcal{P}$ . *Intuitionistic logic* rejects only  $\neg\neg\mathcal{P} \vdash \mathcal{P}$ .
- Why reject double-negation elimination? This rule is in fact equivalent to the law of excluded middle, which says that we always have  $\mathcal{P} \vee \neg\mathcal{P}$ . When we include the rule  $\neg\neg\mathcal{P} \vdash \mathcal{P}$ , we allow the following kind of proof: “If  $\neg\mathcal{P}$  cannot occur, then we have a proof of  $\mathcal{P}$ .” Intuitionists object to this kind of reasoning because it does not provide a constructive proof.
- The rule of explosion  $\perp \vdash \mathcal{P}$  is excluded from some kinds of logic because, on the one hand, it does not give a constructive proof of  $\mathcal{P}$  and, on the other, as soon as a contradiction (even a local one) is found, the rule makes it possible to prove any formula, making all reasoning trivial and therefore unusable.

**3.2. Examples****Example.**

|                                                                                         |  |                                                    |
|-----------------------------------------------------------------------------------------|--|----------------------------------------------------|
| $(\mathcal{P} \wedge \mathcal{Q}) \vee \mathcal{R} \vdash \mathcal{Q} \vee \mathcal{R}$ |  |                                                    |
| $(\mathcal{P} \wedge \mathcal{Q}) \vee \mathcal{R}$                                     |  | Assumption                                         |
| $\mathcal{P} \wedge \mathcal{Q}$                                                        |  | First case of OR                                   |
| $\mathcal{Q}$                                                                           |  | By elimination in $\mathcal{P} \wedge \mathcal{Q}$ |
| $\mathcal{Q} \vee \mathcal{R}$                                                          |  | Since we have $\mathcal{Q}$                        |
| $\mathcal{R}$                                                                           |  | Second case of OR                                  |
| $\mathcal{Q} \vee \mathcal{R}$                                                          |  | Since we have $\mathcal{R}$                        |
| $\vdash \mathcal{Q} \vee \mathcal{R}$                                                   |  | Both cases give $\mathcal{Q} \vee \mathcal{R}$     |

**Example.**

$$\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{S} \vdash \mathcal{R} \vee \mathcal{S}$$

|          |                                       |                                                |
|----------|---------------------------------------|------------------------------------------------|
|          | $\mathcal{P} \vee \mathcal{Q}$        | Assumptions                                    |
|          | $\mathcal{P} \rightarrow \mathcal{R}$ |                                                |
|          | $\mathcal{Q} \rightarrow \mathcal{S}$ |                                                |
|          | <hr/>                                 |                                                |
|          | $\mathcal{P}$                         | First case of OR                               |
|          | $\mathcal{R}$                         | We apply $\mathcal{P} \rightarrow \mathcal{R}$ |
|          | $\mathcal{R} \vee \mathcal{S}$        |                                                |
|          | $\mathcal{Q}$                         | Second case of OR                              |
|          | $\mathcal{S}$                         |                                                |
|          | $\mathcal{R} \vee \mathcal{S}$        |                                                |
| $\vdash$ | $\mathcal{R} \vee \mathcal{S}$        | Both cases lead to the same conclusion         |

**Example.**

$$\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R} \quad \vdash \quad \mathcal{P} \rightarrow (\mathcal{Q} \wedge \mathcal{R})$$

|          |                                                            |                                                                                |
|----------|------------------------------------------------------------|--------------------------------------------------------------------------------|
|          | $\mathcal{P} \rightarrow \mathcal{Q}$                      | Assumptions                                                                    |
|          | $\mathcal{P} \rightarrow \mathcal{R}$                      |                                                                                |
|          | <hr/>                                                      |                                                                                |
|          | $\mathcal{P}$                                              | Temporary assumption                                                           |
|          | $\mathcal{Q}$                                              | Since we have $\mathcal{P}$ and $\mathcal{P} \rightarrow \mathcal{Q}$          |
|          | $\mathcal{R}$                                              | Since we have $\mathcal{P}$ and $\mathcal{P} \rightarrow \mathcal{R}$          |
|          | $\mathcal{Q} \wedge \mathcal{R}$                           |                                                                                |
| $\vdash$ | $\mathcal{P} \rightarrow (\mathcal{Q} \wedge \mathcal{R})$ | Whenever we have $\mathcal{P}$ , we also have $\mathcal{Q} \wedge \mathcal{R}$ |

**Example.**

$$\mathcal{P} \vee \mathcal{Q}, \neg \mathcal{P} \quad \vdash \quad \mathcal{Q}$$

This is an example of the use of explosion: once we have proved  $\perp$ , we can prove anything.

|          |                                |                                                             |
|----------|--------------------------------|-------------------------------------------------------------|
|          | $\mathcal{P} \vee \mathcal{Q}$ |                                                             |
|          | $\neg \mathcal{P}$             |                                                             |
|          | <hr/>                          |                                                             |
|          | $\mathcal{P}$                  | First case of OR                                            |
|          | $\perp$                        | Contradiction: we have $\mathcal{P}$ and $\neg \mathcal{P}$ |
|          | $\mathcal{Q}$                  | Anything follows by explosion                               |
|          | $\mathcal{Q}$                  | Second case of OR                                           |
|          | $\mathcal{Q}$                  |                                                             |
| $\vdash$ | $\mathcal{Q}$                  | Conclusion from both cases                                  |

### 3.3. Derived rules

From the basic rules of inference, we derive a limited number of other rules. These new rules make proofs easier to write.

#### Derived rules. Equivalence.

As expected, we prove an equivalence by proving both implications, and conversely.

$$\boxed{\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{P} \rightarrow \mathcal{Q} \quad \text{and} \quad \mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{Q} \rightarrow \mathcal{P}}$$

$$\boxed{\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{Q} \rightarrow \mathcal{P} \vdash \mathcal{P} \leftrightarrow \mathcal{Q}}$$

#### Derived rules. Negation.

First, here is the definition of negation in terms of implication and the always-false formula  $\perp$ .

$$\boxed{\neg \mathcal{P} \text{ is defined as } \mathcal{P} \rightarrow \perp}$$

Thus, by definition, we have the two rules:

$$\boxed{\neg \mathcal{P} \vdash \mathcal{P} \rightarrow \perp \quad \text{and} \quad \mathcal{P} \rightarrow \perp \vdash \neg \mathcal{P}}$$

Thus, if we have  $\mathcal{P}$  and  $\neg \mathcal{P}$ , then implication gives us a contradiction:

$$\boxed{\mathcal{P}, \neg \mathcal{P} \vdash \perp}$$

Of course,  $\neg \top$  is  $\perp$ , and conversely  $\neg \perp$  is  $\top$ .

We also have the converse direction of the double-negation rule:

$$\boxed{\mathcal{P} \vdash \neg \neg \mathcal{P}}$$

We also have the law of excluded middle:

$$\boxed{\vdash \mathcal{P} \vee \neg \mathcal{P}}$$

That is, without any assumption, we always have a proof of  $\mathcal{P} \vee \neg\mathcal{P}$ .

### Derived rules. Conjunction/disjunction.

For example:

$$\boxed{\mathcal{P} \vee \mathcal{Q}, \neg\mathcal{Q} \vdash \mathcal{P}}$$

We can also consider all the proofs related to De Morgan's laws, for example:

$$\boxed{\neg(\mathcal{P} \wedge \mathcal{Q}) \vdash \neg\mathcal{P} \vee \neg\mathcal{Q}}$$

We will see how to derive this rule from the basic rules in the examples below.

### Derived rules. Implication.

First, here is the rule called *modus tollens*:

$$\boxed{\mathcal{P} \rightarrow \mathcal{Q}, \neg\mathcal{Q} \vdash \neg\mathcal{P}}$$

If  $\mathcal{P}$  implies  $\mathcal{Q}$ , but we do not have  $\mathcal{Q}$ , then we have a proof of  $\neg\mathcal{P}$ . The examples below explain how to derive this rule from the basic rules.

The law of excluded middle allows a proof by cases:

$$\boxed{\mathcal{P} \rightarrow \mathcal{Q}, \neg\mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}}$$

This allows us to reason using the two cases of  $\mathcal{P} \vee \neg\mathcal{P}$  (whatever the formula  $\mathcal{P}$ ).

$$\begin{array}{c|c|c} & \mathcal{P} & \text{First case} \\ & \hline & \mathcal{Q} & \\ & \neg\mathcal{P} & \text{Second case} \\ & \hline & \mathcal{Q} & \\ \vdash & \mathcal{Q} & \end{array}$$

### Derived rules. Proof by contradiction and contraposition.

Proof by contradiction:

$$\boxed{\neg\mathcal{P} \rightarrow \perp \vdash \mathcal{P}}$$

If  $\neg\mathcal{P}$  leads to a contradiction, then we have proved  $\mathcal{P}$ .

Contraposition:

$$\boxed{\mathcal{P} \rightarrow \mathcal{Q} \vdash \neg\mathcal{Q} \rightarrow \neg\mathcal{P}}$$

Or, equivalently:

$$\boxed{\neg\mathcal{Q} \rightarrow \neg\mathcal{P} \vdash \mathcal{P} \rightarrow \mathcal{Q}}$$

### 3.4. Examples

**Example.**

|          |                                                                                                                                                                                                                                                                   |                                                                                                                                      |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
|          |                                                                                                                                                                                                                                                                   | $\mathcal{P} \vee \mathcal{Q} \rightarrow \mathcal{P} \wedge \mathcal{Q} \quad \vdash \quad \mathcal{P} \leftrightarrow \mathcal{Q}$ |
| $\vdash$ | $\mathcal{P} \vee \mathcal{Q} \rightarrow \mathcal{P} \wedge \mathcal{Q}$                                                                                                                                                                                         |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P}</math></div> <div>Assume <math>\mathcal{P}</math></div> </div>                                 |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P} \vee \mathcal{Q}</math></div> <div></div> </div>                                               |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P} \wedge \mathcal{Q}</math></div> <div>By the initial assumption</div> </div>                    |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{Q}</math></div> <div></div> </div>                                                                |                                                                                                                                      |
|          | $\mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                                                                                                                             | First direction of the equivalence                                                                                                   |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{Q}</math></div> <div>Assume <math>\mathcal{Q}</math></div> </div>                                 |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P} \vee \mathcal{Q}</math></div> <div></div> </div>                                               |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P} \wedge \mathcal{Q}</math></div> <div>By the implication in the initial assumption</div> </div> |                                                                                                                                      |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P}</math></div> <div></div> </div>                                                                |                                                                                                                                      |
|          | $\mathcal{Q} \rightarrow \mathcal{P}$                                                                                                                                                                                                                             | Second direction of the equivalence                                                                                                  |
|          | $\mathcal{P} \leftrightarrow \mathcal{Q}$                                                                                                                                                                                                                         |                                                                                                                                      |

**Example.**

Let us prove *modus tollens* directly by contradiction:

|          |                                                                                                                                                                                                                                                                               |                                                                                             |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
|          |                                                                                                                                                                                                                                                                               | $\mathcal{P} \rightarrow \mathcal{Q}, \neg \mathcal{Q} \quad \vdash \quad \neg \mathcal{P}$ |
| $\vdash$ | $\mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                                                                                                                                         |                                                                                             |
|          | $\neg \mathcal{Q}$                                                                                                                                                                                                                                                            |                                                                                             |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{P}</math></div> <div>For a contradiction, assume <math>\mathcal{P}</math></div> </div>                        |                                                                                             |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\mathcal{Q}</math></div> <div>The initial assumption gives <math>\mathcal{Q}</math></div> </div>                       |                                                                                             |
|          | <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="border-left: 1px solid black; padding-left: 10px;"><math>\perp</math></div> <div>Contradiction: we have <math>\mathcal{Q}</math> and <math>\neg \mathcal{Q}</math></div> </div> |                                                                                             |
|          | $\mathcal{P} \rightarrow \perp$                                                                                                                                                                                                                                               |                                                                                             |
|          | $\neg \mathcal{P}$                                                                                                                                                                                                                                                            |                                                                                             |
|          |                                                                                                                                                                                                                                                                               |                                                                                             |
|          |                                                                                                                                                                                                                                                                               |                                                                                             |
|          |                                                                                                                                                                                                                                                                               |                                                                                             |

**Example.**

Let us prove the following De Morgan rule directly:

$$\neg(\mathcal{P} \wedge \mathcal{Q}) \quad \vdash \quad \neg \mathcal{P} \vee \neg \mathcal{Q}$$

|          |                                          |                                                      |
|----------|------------------------------------------|------------------------------------------------------|
|          | $\neg(\mathcal{P} \wedge \mathcal{Q})$   |                                                      |
|          | $\mathcal{P}$                            | First case: assume $\mathcal{P}$                     |
|          | $\mathcal{Q}$                            | For a contradiction, assume $\mathcal{Q}$            |
|          | $\mathcal{P} \wedge \mathcal{Q}$         |                                                      |
|          | $\perp$                                  | Contradiction, obtained from the initial assumption  |
|          | $\neg \mathcal{Q}$                       |                                                      |
|          | $\neg \mathcal{P} \vee \neg \mathcal{Q}$ |                                                      |
|          | $\neg \mathcal{P}$                       | Second case: assume $\neg \mathcal{P}$               |
|          | $\neg \mathcal{P} \vee \neg \mathcal{Q}$ |                                                      |
| $\vdash$ | $\neg \mathcal{P} \vee \neg \mathcal{Q}$ | Conclusion obtained in both cases of excluded middle |

### 3.5. Proof by contradiction and contraposition

In classical logic, to prove an implication  $\mathcal{P} \rightarrow \mathcal{Q}$ , we can replace a proof by contraposition with a proof by contradiction, and conversely.

The two arguments in fact use essentially the same steps.

#### From contraposition to contradiction.

Suppose that we know how to prove the contrapositive  $\neg \mathcal{Q} \rightarrow \neg \mathcal{P}$ . Such a proof has the form

$$\neg \mathcal{Q} \longrightarrow \boxed{\mathcal{C}} \longrightarrow \neg \mathcal{P},$$

where  $\mathcal{C}$  denotes the intermediate steps of the argument (the proof on the left below).

To obtain a proof by contradiction of  $\mathcal{P} \rightarrow \mathcal{Q}$ , we assume  $\mathcal{P}$  and then  $\neg \mathcal{Q}$ . We repeat exactly the same argument  $\mathcal{C}$ , which yields  $\neg \mathcal{P}$ , contradicting  $\mathcal{P}$  (the proof on the right below).

|          |                       |                    |          |                                       |            |
|----------|-----------------------|--------------------|----------|---------------------------------------|------------|
|          | $\neg \mathcal{Q}$    |                    |          | $\mathcal{P} \wedge \neg \mathcal{Q}$ |            |
|          | $\boxed{\mathcal{C}}$ | Intermediate steps |          | $\boxed{\mathcal{C}}$                 | Same steps |
|          | $\neg \mathcal{P}$    |                    |          | $\mathcal{P} \wedge \neg \mathcal{P}$ |            |
| $\vdash$ |                       |                    | $\vdash$ | $\perp$                               |            |

#### From contradiction to contraposition.

Now suppose that we have a proof by contradiction of  $\mathcal{P} \rightarrow \mathcal{Q}$  (on the left below). It has the form:

$$\mathcal{P}, \neg \mathcal{Q} \longrightarrow \boxed{\mathcal{C}'} \longrightarrow \perp.$$

To obtain the contrapositive, we assume  $\neg \mathcal{Q}$  and seek to prove  $\neg \mathcal{P}$ . We therefore assume  $\mathcal{P}$  and repeat exactly the same argument  $\mathcal{C}'$ , which leads to a contradiction (on the right below).

|                                                                                                                                  |                    |                                                                                                                                                                                       |            |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| $\begin{array}{c c} & \mathcal{P} \wedge \neg \mathcal{Q} \\ \hline & \boxed{\mathcal{C}'} \\ \hline \vdash & \perp \end{array}$ | Intermediate steps | $\begin{array}{c c c} & \neg \mathcal{Q} & \\ \hline & \mathcal{P} & \\ \hline & \boxed{\mathcal{C}'} & \\ \hline \vdash & \perp & \\ \hline \vdash & \neg \mathcal{P} & \end{array}$ | Same steps |
|----------------------------------------------------------------------------------------------------------------------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|

Thus, a proof by contraposition and a proof by contradiction have the same core: only the arrangement of the assumptions changes.

### 3.6. Summary

Here are the main rules of inference used in this chapter, with the basic rules of classical logic in bold.

| Name                                         | Rule                                                                                                                                                                                                                                                                                                      |
|----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Reuse</b>                                 | $\mathcal{P} \vdash \mathcal{P}$                                                                                                                                                                                                                                                                          |
| <b><math>\wedge</math> elimination</b>       | $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$<br>$\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}$                                                                                                                                                                                                |
| <b><math>\wedge</math> introduction</b>      | $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$                                                                                                                                                                                                                                          |
| <b>Double-negation elimination</b>           | $\neg \neg \mathcal{P} \vdash \mathcal{P}$                                                                                                                                                                                                                                                                |
| <b><math>\vee</math> introduction</b>        | $\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q}$<br>$\mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}$                                                                                                                                                                                                    |
| <b><math>\vee</math> elimination</b>         | $\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{R} \vdash \mathcal{R}$                                                                                                                                                                               |
| <b>Implication (modus ponens)</b>            | $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}$                                                                                                                                                                                                                                     |
| <b><math>\rightarrow</math> introduction</b> | If $\mathcal{P} \vdash \mathcal{Q}$ then $\vdash \mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                                                                                                                     |
| <b>Explosion (ex falso)</b>                  | $\perp \vdash \mathcal{P}$                                                                                                                                                                                                                                                                                |
| <i>Equivalence</i>                           | $\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{P} \rightarrow \mathcal{Q}$<br>$\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{Q} \rightarrow \mathcal{P}$<br>$\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{Q} \rightarrow \mathcal{P} \vdash \mathcal{P} \leftrightarrow \mathcal{Q}$ |
| <i>Definition of negation</i>                | $\neg \mathcal{P}$ is defined as $\mathcal{P} \rightarrow \perp$                                                                                                                                                                                                                                          |
| <i>Double-negation introduction</i>          | $\mathcal{P} \vdash \neg \neg \mathcal{P}$                                                                                                                                                                                                                                                                |
| <i>Law of excluded middle</i>                | $\vdash \mathcal{P} \vee \neg \mathcal{P}$                                                                                                                                                                                                                                                                |
| <i>Modus tollens</i>                         | $\mathcal{P} \rightarrow \mathcal{Q}, \neg \mathcal{Q} \vdash \neg \mathcal{P}$                                                                                                                                                                                                                           |
| <i>De Morgan's laws</i>                      | $\neg(\mathcal{P} \wedge \mathcal{Q}) \vdash \neg \mathcal{P} \vee \neg \mathcal{Q}$<br>...                                                                                                                                                                                                               |
| <i>Proof by contradiction</i>                | $\neg \mathcal{P} \rightarrow \perp \vdash \mathcal{P}$                                                                                                                                                                                                                                                   |
| <i>Contrapositive</i>                        | $\mathcal{P} \rightarrow \mathcal{Q} \vdash \neg \mathcal{Q} \rightarrow \neg \mathcal{P}$<br>$\neg \mathcal{Q} \rightarrow \neg \mathcal{P} \vdash \mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                  |

## 4. An overview of Gödel's Completeness Theorem

Gödel's *Incompleteness Theorem* (1931) is the ultimate goal of this book. This theorem will shake the foundations of logic. But to truly grasp its importance, we must first understand the *Completeness Theorem* (1929), which is rather reassuring. Its statement even seems ordinary, because it is exactly what everyone would expect.

**Theorem 1** (Completeness Theorem).

For true/false logic:

$$\mathcal{P} \models \mathcal{Q} \quad \text{if and only if} \quad \mathcal{P} \vdash \mathcal{Q}.$$

The key point is that we are working here in the framework of propositional logic, presented in the chapter “True/False Logic.” This is also true in the more general framework of first-order logic (see the chapter “First-Order Logic”). Indeed, that is the framework in which we will prove this theorem (see the chapter “Completeness Theorem”).

Let us examine each implication in the equivalence stated above.

Reverse direction  $\Leftarrow$ .  $\mathcal{P} \vdash \mathcal{Q}$  implies  $\mathcal{P} \models \mathcal{Q}$ .

This implication is called the *Soundness Theorem*; it is a statement of “common sense.” If we have a proof that  $\mathcal{P}$  entails  $\mathcal{Q}$  and  $\mathcal{P}$  is true, then  $\mathcal{Q}$  is true. In other words, when we have proved a theorem and its assumptions are true, then its conclusion is true! Note once again that the formal definition of a proof contains no concept of true or false. The Soundness Theorem means that the rules of inference chosen to define a proof are coherent and compatible with the rules of propositional logic.

Forward direction  $\Rightarrow$ .  $\mathcal{P} \models \mathcal{Q}$  implies  $\mathcal{P} \vdash \mathcal{Q}$ .

This direction is the *Completeness Theorem*. It means that for every statement we find to be true, we can find a proof that makes it a theorem. Thus, if we find that whenever  $\mathcal{P}$  is true,  $\mathcal{Q}$  is also true, then there is a proof that  $\mathcal{P}$  entails  $\mathcal{Q}$ . This result holds within propositional logic (our true/false logic).

This gives us sweet dreams: in propositional logic, everything that is true is provable, and conversely. The nightmare will begin later in this book...

## Exercises

### Proof by truth table

**Exercise 1.** Determine whether the following logical consequences are valid. Justify your answer.

|                                                                                                           |                         |                                                               |
|-----------------------------------------------------------------------------------------------------------|-------------------------|---------------------------------------------------------------|
| $(\mathcal{P} \wedge \neg \mathcal{Q}) \vee (\neg \mathcal{P} \wedge \mathcal{Q})$                        | $\stackrel{?}{\models}$ | $\mathcal{P} \vee \mathcal{Q}$                                |
| $\mathcal{P} \rightarrow \mathcal{Q}, \neg \mathcal{P}$                                                   | $\stackrel{?}{\models}$ | $\neg \mathcal{Q}$                                            |
| $\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{R} \rightarrow \mathcal{Q}$                                | $\stackrel{?}{\models}$ | $\mathcal{P} \rightarrow \mathcal{R}$                         |
| $\mathcal{P} \rightarrow \neg \mathcal{Q}$                                                                | $\stackrel{?}{\models}$ | $\mathcal{Q} \rightarrow \neg \mathcal{P}$                    |
| $\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{P} \vee \neg \mathcal{Q}$                                  | $\stackrel{?}{\models}$ | $\mathcal{Q}$                                                 |
| $\mathcal{P} \vee \neg \mathcal{Q}, \mathcal{P} \vee \neg \mathcal{R}, \mathcal{R} \vee \neg \mathcal{Q}$ | $\stackrel{?}{\models}$ | $\neg \mathcal{P} \rightarrow \mathcal{Q} \wedge \mathcal{R}$ |

## Formal proof

**Exercise 2.** Explain the following theorems in English:

$$\begin{array}{rcl}
 \mathcal{P}, \mathcal{Q} & \vdash & \mathcal{P} \wedge \mathcal{Q} \\
 \mathcal{P} \wedge (\mathcal{P} \rightarrow \mathcal{Q}) & \vdash & \mathcal{Q} \\
 \mathcal{P} \rightarrow \mathcal{Q}, \mathcal{Q} \rightarrow \mathcal{R} & \vdash & \mathcal{P} \rightarrow \mathcal{R} \\
 \neg \mathcal{Q} \rightarrow \neg \mathcal{P} & \vdash & \mathcal{P} \rightarrow \mathcal{Q} \\
 & \vdash & \mathcal{P} \vee \neg \mathcal{P}
 \end{array}$$

**Exercise 3.** Consider the following theorem:

$$\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{S} \quad \vdash \quad \mathcal{R} \vee \mathcal{S}$$

Complete the missing lines of the proof and comment on every line.

|          |               |                                        |
|----------|---------------|----------------------------------------|
|          | ...           | Assumption 1                           |
|          | ...           | ...                                    |
|          | ...           | ...                                    |
|          |               |                                        |
|          | $\mathcal{P}$ | Temporary assumption. First case of OR |
|          | ...           | We used the implication ...            |
|          | ...           | Intermediate conclusion                |
|          | ...           | Temporary assumption. ...              |
|          | ...           | ...                                    |
|          | ...           | ...                                    |
| $\vdash$ | ...           | Conclusion because ...                 |

## Rules of inference

**Exercise 4.** Give natural deduction proofs of the following theorems:

$$\begin{array}{rcl}
 \mathcal{P} \rightarrow \mathcal{Q}, \mathcal{Q} \rightarrow \mathcal{R} & \vdash & \mathcal{P} \rightarrow \mathcal{R} \\
 \mathcal{P} & \vdash & (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \neg \mathcal{Q}) \\
 (\mathcal{P} \rightarrow \mathcal{Q}) \rightarrow \mathcal{R} & \vdash & \mathcal{P} \rightarrow (\mathcal{Q} \rightarrow \mathcal{R}) \\
 \mathcal{P} \leftrightarrow \mathcal{Q} & \vdash & (\mathcal{P} \vee \mathcal{Q}) \rightarrow (\mathcal{P} \wedge \mathcal{Q}) \\
 \neg \mathcal{P} \vee \neg \mathcal{Q} & \vdash & \neg(\mathcal{P} \wedge \mathcal{Q})
 \end{array}$$

**Exercise 5.** Give natural deduction proofs of the following theorems:

$$\begin{array}{rcl}
 \neg \mathcal{B} \rightarrow (\mathcal{A} \vee \mathcal{B}) & \vdash & \mathcal{A} \\
 \mathcal{C} \rightarrow (\mathcal{D} \rightarrow \mathcal{E}) & \vdash & (\mathcal{C} \wedge \mathcal{D}) \rightarrow \mathcal{E} \\
 \mathcal{F} \leftrightarrow \mathcal{G}, \mathcal{G} \leftrightarrow \mathcal{H} & \vdash & \mathcal{F} \leftrightarrow \mathcal{H} \\
 & \vdash & (\mathcal{I} \rightarrow \mathcal{J}) \vee (\mathcal{J} \rightarrow \mathcal{I}) \\
 \mathcal{K} \vee \mathcal{L}, \mathcal{L} \vee \mathcal{M}, \neg \mathcal{L} & \vdash & \mathcal{K} \wedge \mathcal{M}
 \end{array}$$

**Exercise 6.** The aim of this exercise is to show that the law of excluded middle, “ $\vdash \mathcal{P} \vee \neg \mathcal{P}$ ” (with no assumptions), follows from the basic rules. Recall that, by definition,  $\neg \mathcal{P}$  means  $\mathcal{P} \rightarrow \perp$ .

1. Check that  $\neg \neg \mathcal{Q}$  can be written  $(\mathcal{Q} \rightarrow \perp) \rightarrow \perp$ .

2. Here is the proof of:

$$\vdash ((\mathcal{P} \vee \neg \mathcal{P}) \rightarrow \perp) \rightarrow \perp$$

$$\begin{array}{c}
 \vdash \left| \begin{array}{c}
 \overline{\emptyset} \\
 (\mathcal{P} \vee \neg \mathcal{P}) \rightarrow \perp \\
 \left| \begin{array}{c}
 \mathcal{P} \\
 \hline
 \mathcal{P} \vee \neg \mathcal{P} \\
 \perp \\
 \mathcal{P} \rightarrow \perp \\
 \neg \mathcal{P} \\
 \mathcal{P} \vee \neg \mathcal{P} \\
 \perp
 \end{array}
 \right. \\
 ((\mathcal{P} \vee \neg \mathcal{P}) \rightarrow \perp) \rightarrow \perp
 \end{array} \right.
 \end{array}$$

Explain each line and check that only the basic rules are used.

3. Conclude using a basic rule.

**Exercise 7.** The aim is to obtain derived rules from the basic rules.

1. Prove the derived rule

$$\mathcal{P} \vdash \neg \neg \mathcal{P}$$

using only the basic rules. Recall that  $\neg \mathcal{P}$  means  $\mathcal{P} \rightarrow \perp$ .

2. Here we allow the basic rules and the law of excluded middle (which we derived from the basic rules). We may therefore reason by distinguishing the cases  $\mathcal{P}$  and  $\neg \mathcal{P}$ . Prove the rule for proof by contradiction:

$$\neg \mathcal{P} \rightarrow \perp \vdash \mathcal{P}$$

3. Deduce that this form of contraposition can be derived using only the basic rules:

$$\mathcal{P} \rightarrow \mathcal{Q} \vdash \neg \mathcal{Q} \rightarrow \neg \mathcal{P}$$

## An overview of Gödel's Completeness Theorem

**Exercise 8.** Show that both

$$\neg(\mathcal{P} \vee \mathcal{Q}) \models (\neg \mathcal{P}) \wedge (\neg \mathcal{Q})$$

and

$$\neg(\mathcal{P} \vee \mathcal{Q}) \vdash (\neg \mathcal{P}) \wedge (\neg \mathcal{Q})$$

hold.

**Exercise 9.** Show that both

$$\neg \mathcal{P} \vee \mathcal{Q}, \neg \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{R} \models \mathcal{R}$$

and

$$\neg \mathcal{P} \vee \mathcal{Q}, \neg \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{R} \vdash \mathcal{R}$$

hold.

# Which Connectives Are Useful?

The connectives  $\neg$  (NOT),  $\wedge$  (AND), and  $\vee$  (OR) are sufficient to construct every formula in true/false logic.

## 1. Evaluating formulas

### 1.1. Review

Recall that a *formula* is a finite word constructed from:

- a set  $\text{Var} = \{P_1, P_2, \dots, P_n\}$  of propositional variables, where  $n \geq 1$ ,
- parentheses ( and ),
- and a set  $\text{Conn}$  of logical connectives.

So far, our choice of logical connectives has been:

$$\text{Conn} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$$

Is this a sensible choice? Is it sufficient to express complex phenomena? To answer these questions, we will model the evaluation of a formula as the evaluation of a function whose inputs and output take only the values **0**/**1**.

Thus, in this chapter, the Boolean values are  $\mathbb{B} = \{0, 1\}$ , with **1** for T and **0** for F.

### 1.2. Abstract functions from $\{0, 1\}^n$ to $\{0, 1\}$

Let  $n \geq 1$  be a fixed integer. An *abstract function* is a function

$$f : \{0, 1\}^n \rightarrow \{0, 1\}$$

Let  $x = (\varepsilon_1, \dots, \varepsilon_n) \in \{0, 1\}^n$ . It is a vector whose coordinates are all **0** or **1**. In addition,  $f(x)$  is either **0** or **1**.

**Example.**

Let  $n = 3$ . Define a function  $f : \{0, 1\}^3 \rightarrow \{0, 1\}$  by its  $2^n = 8$  values.

| $x$         | $f(x)$ |
|-------------|--------|
| $(0, 0, 0)$ | 0      |
| $(0, 0, 1)$ | 0      |
| $(0, 1, 0)$ | 1      |
| $(0, 1, 1)$ | 1      |
| $(1, 0, 0)$ | 0      |
| $(1, 0, 1)$ | 1      |
| $(1, 1, 0)$ | 0      |
| $(1, 1, 1)$ | 1      |

**Review.** Let  $E$  and  $F$  be two finite sets. The total number of distinct functions  $f : E \rightarrow F$  is  $(\text{Card } F)^{\text{Card } E}$ . Here  $\text{Card } E = \text{Card}\{0, 1\}^n = 2^n$  and  $\text{Card } F = \text{Card}\{0, 1\} = 2$ . Thus the total number of abstract functions  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  is  $2^{(2^n)}$ .

### 1.3. Functions from a formula

We will now construct a function  $f_{\mathcal{P}} : \{0, 1\}^n \rightarrow \{0, 1\}$  from a formula.

Let  $\mathcal{P}$  be a formula expressed in terms of the  $n$  propositional variables  $P_1, \dots, P_n$ . The **evaluation** of  $\mathcal{P}$  at  $x = (\varepsilon_1, \dots, \varepsilon_n) \in \{0, 1\}^n$  consists of assigning the value  $\varepsilon_1$  (0 or 1) to  $P_1$ , the value  $\varepsilon_2$  to  $P_2$ , and so on. The result of this evaluation is a value 0 or 1.

Note: A rigorous construction of this evaluation function will be given in the chapter “Compactness Theorem.”

The evaluation of  $\mathcal{P}$  therefore corresponds to a function:

$$\begin{aligned} f_{\mathcal{P}} : \{0, 1\}^n &\longrightarrow \{0, 1\} \\ x = (\varepsilon_1, \dots, \varepsilon_n) &\longmapsto f_{\mathcal{P}}(\varepsilon_1, \dots, \varepsilon_n) \end{aligned}$$

This is another way of presenting the truth table of the formula  $\mathcal{P}$ .

#### Example.

Let  $\mathcal{P}$  be the following formula:

$$\mathcal{P} : (P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)$$

We obtain a function  $f_{\mathcal{P}} : \{0, 1\}^3 \rightarrow \{0, 1\}$ , which we compute one value at a time. For example, at  $x = (0, 1, 0)$  we assign (F, T, F) to the variables  $(P_1, P_2, P_3)$ . The formula  $\mathcal{P}$  then evaluates to  $(F \vee T) \wedge (\neg F \vee F)$ , and hence to T. Thus  $f_{\mathcal{P}}(0, 1, 0) = 1$ .

| $x$         | $f_{\mathcal{P}}(x)$ |
|-------------|----------------------|
| $(0, 0, 0)$ | 0                    |
| $(0, 0, 1)$ | 0                    |
| $(0, 1, 0)$ | 1                    |
| $(0, 1, 1)$ | 1                    |
| $(1, 0, 0)$ | 0                    |
| $(1, 0, 1)$ | 1                    |
| $(1, 1, 0)$ | 0                    |
| $(1, 1, 1)$ | 1                    |

Here we observe that our function  $f_{\mathcal{P}}$  is exactly the same as the abstract function  $f$  defined in the

previous example.

## 1.4. Can every function be represented?

In the previous example, we had a formula  $\mathcal{P}$  whose associated function  $f_{\mathcal{P}}$  was the same as an abstract function defined earlier. But is this always possible? Given an abstract function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , can we find a formula  $\mathcal{P}$  such that  $f_{\mathcal{P}}$  is exactly the function  $f$ ?

If the answer were no, it would mean that our connectives are not sufficient to model every function and that we would have to invent new connectives to represent some functions. The question is all the more pressing because, when  $n$  is large, the number of functions  $f$  to represent is  $2^{2^n}$ , which is very large indeed.

## 1.5. Unary connectives

Before turning to a general theoretical result, let us begin by studying the cases  $n = 1$  and  $n = 2$  by hand.

For  $n = 1$ , there are  $4 = 2^{2^1}$  functions  $f : \{0, 1\} \rightarrow \{0, 1\}$ . Here are the 4 possible truth tables:

|   | $\top$ |   | $\perp$ |   | id |   | $\neg$ |
|---|--------|---|---------|---|----|---|--------|
| 0 | 1      | 0 | 0       | 0 | 0  | 0 | 1      |
| 1 | 1      | 1 | 0       | 1 | 1  | 1 | 0      |

These 4 functions are  $\top$  (always true),  $\perp$  (always false), id (identity), and  $\neg$  (negation).

The case  $n = 1$  is summarized in a single table.

| Symbol  | Representation    | Name                         |
|---------|-------------------|------------------------------|
| $\top$  | $P \vee \neg P$   | tautology / always true      |
| $\perp$ | $P \wedge \neg P$ | contradiction / always false |
| id      | $P$               | identity                     |
| $\neg$  | $\neg P$          | negation                     |

The second column gives a representation by a formula using the connectives  $\neg$ ,  $\wedge$ , and  $\vee$ .

## 1.6. Binary connectives

For  $n = 2$ , there are  $16 = 2^{2^2}$  functions  $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ . Below is a list of all these functions and their representations by formulas. The second column gives a representation by a formula using the connectives  $\neg$ ,  $\wedge$ ,  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$ . The last column gives the 4 values of the function when  $(P, Q)$  is successively  $(0, 0)$ ,  $(0, 1)$ ,  $(1, 0)$ , and  $(1, 1)$ .

| Symbol            | Representation                       | Name                         | Values |
|-------------------|--------------------------------------|------------------------------|--------|
| $\perp$           | $P \wedge \neg P$                    | always false / contradiction | 0000   |
| $\wedge$          | $P \wedge Q$                         | AND/ conjunction             | 0001   |
|                   | $P \wedge \neg Q$                    | $P$ and (not $Q$ )           | 0010   |
|                   | $P$                                  | left projection              | 0011   |
|                   | $\neg P \wedge Q$                    | (not $P$ ) and $Q$           | 0100   |
|                   | $Q$                                  | right projection             | 0101   |
| $\oplus$          | $(P \vee Q) \wedge \neg(P \wedge Q)$ | XOR / exclusive or           | 0110   |
| $\vee$            | $P \vee Q$                           | OR/ disjunction              | 0111   |
| $\downarrow$      | $\neg(P \vee Q)$                     | NOR                          | 1000   |
| $\leftrightarrow$ | $P \leftrightarrow Q$                | equivalence                  | 1001   |
|                   | $\neg Q$                             | negation of $Q$              | 1010   |
|                   | $Q \rightarrow P$                    | $Q$ implies $P$              | 1011   |
|                   | $\neg P$                             | negation of $P$              | 1100   |
| $\rightarrow$     | $P \rightarrow Q$                    | $P$ implies $Q$              | 1101   |
| $\uparrow$        | $\neg(P \wedge Q)$                   | NAND / Sheffer               | 1110   |
| $\top$            | $P \vee \neg P$                      | always true / tautology      | 1111   |

For example, the connective  $\downarrow$ /NOR satisfies  $0 \downarrow 0 = 1$ ,  $0 \downarrow 1 = 0$ ,  $1 \downarrow 0 = 0$ ,  $1 \downarrow 1 = 0$ . But  $P \downarrow Q$  is in fact represented by the formula  $\neg(P \vee Q)$ . Thus we do not really need a special symbol for this operation. However, two new connectives are useful in practice: XOR and NAND.

- The **exclusive or**, denoted by  $\oplus$ /XOR, is true when one of the terms is true, but not both. It corresponds to addition  $x + y$  modulo 2.
- The connective  $\uparrow$ /NAND, also called the Sheffer stroke, is represented as the negation of AND:  $P \uparrow Q = \neg(P \wedge Q)$ .

## 2. A complete set of connectives

### 2.1. Theorem

#### Theorem 1.

The set  $\{\neg, \wedge\}$  consisting of the connectives NOT and AND forms a complete set of connectives.

We will see the proof in the next section. This means that, for every  $n \geq 1$  and every  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ , there is a formula  $\mathcal{P}$  involving only the connectives  $\neg$  and  $\wedge$  such that the function  $f_{\mathcal{P}}$  arising from  $\mathcal{P}$  is exactly  $f$ . In other words,  $f$  and  $f_{\mathcal{P}}$  have the same truth table.

The important point to remember is that a finite number of connectives can represent every conceivable Boolean function, whatever  $n$  may be.

Another way to understand this theorem is to say that, for any formula (with any connectives), we can find a logically equivalent formula containing only the connectives  $\neg$  and  $\wedge$ . Recall that two formulas  $\mathcal{P}$  and  $\mathcal{Q}$  are logically equivalent (written  $\mathcal{P} \equiv \mathcal{Q}$ ) if they have the same truth table.

**Not such a silly question!** If we need only  $\{\neg, \wedge\}$  to represent all our functions, why do we add the other connectives  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$ ? Quite simply for practical reasons: to make formulas shorter and, above all, to make their logical meaning easier to understand.

## 2.2. Examples

1. For  $n = 1$  and  $n = 2$ , we have already seen how to represent every function (allowing ourselves to use  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$  as well).
2. The same function can be represented by different formulas. For example,  $P \wedge \neg Q$ ,  $\neg Q \wedge P \wedge \neg Q$ , and  $(\neg \neg P) \wedge (\neg Q)$  represent the same function. The representation is even less unique if all the connectives  $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$  are allowed. For example,  $P \rightarrow Q$ ,  $\neg P \vee Q$ , and  $\neg(P \wedge \neg Q)$  are formulas that represent the same function.
3. Let  $n = 3$ , and let  $f$  be the function that equals **1** at  $(0, 0, 0)$  and  $(1, 1, 1)$  and equals **0** everywhere else. This function is represented by the formula

$$\mathcal{P} : \quad (\neg P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge R)$$

Problem: the formula contains a  $\vee$  (OR), whereas we would like to use only  $\neg$  and  $\wedge$ . We will see the solution in the next subsection.

## 2.3. The usual connectives from $\neg$ and $\wedge$

Since the connectives  $\neg$  and  $\wedge$  are sufficient, they must allow us to recover our other favorite connectives:  $\vee$ ,  $\rightarrow$ , and  $\leftrightarrow$ .

Indeed:

$$\begin{aligned} P \vee Q &\equiv \neg(\neg P \wedge \neg Q) \\ P \rightarrow Q &\equiv \neg P \vee Q \equiv \neg(P \wedge \neg Q) \\ P \leftrightarrow Q &\equiv (P \rightarrow Q) \wedge (Q \rightarrow P) \equiv (\neg(P \wedge \neg Q)) \wedge (\neg(Q \wedge \neg P)) \end{aligned}$$

Recall that the notation  $\equiv$  means that the formulas have exactly the same truth tables. Of course, these relations also hold for formulas  $\mathcal{P}$  and  $\mathcal{Q}$  (in place of the variables  $P$  and  $Q$ ).

Thus, every formula expressed using the connectives  $\mathcal{C} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$  can be rewritten as a logically equivalent formula containing only the connectives  $\mathcal{C}_0 = \{\neg, \wedge\}$ .

## 2.4. Other complete sets of connectives

The set  $\mathcal{C}_1 = \{\neg, \rightarrow\}$  is also a complete set of connectives. The proof follows easily from the theorem: it is enough to show how to generate the connective  $\wedge$  from  $\neg$  and  $\rightarrow$ :

$$P \wedge Q \equiv \neg(P \rightarrow \neg Q)$$

Thus, given any formula, the theorem gives us a logically equivalent formula using only  $\neg$  and  $\wedge$ . We can then replace each  $\wedge$  using the logical equivalence above to obtain a formula containing only  $\neg$  and  $\rightarrow$ .

Even more remarkably, a single connective can generate all of them!  $\mathcal{C}_2 = \{\uparrow\}$  is a complete set. Recall that  $\uparrow$  / NAND is the connective corresponding to the negation of AND:

$$P \uparrow Q \equiv \neg(P \wedge Q)$$

Suppose that the connective  $\uparrow$  is defined. We can then recover  $\neg$  and  $\wedge$ :

$$\neg P \equiv P \uparrow P \quad P \wedge Q \equiv \neg(P \uparrow Q) \equiv (P \uparrow Q) \uparrow (P \uparrow Q)$$

Since  $\uparrow$  can be used to reconstruct both  $\neg$  and  $\wedge$ , and  $\{\neg, \wedge\}$  is complete, it follows that  $\{\uparrow\}$  is a complete set.

From a computer science perspective, this means that a single type of logic gate, the NAND gate, can model any Boolean function.

### 3. Proof

This section is devoted to the proof of the theorem. We will prove that  $\mathcal{C} = \{\neg, \wedge, \vee\}$  is a complete set of connectives. Since we have seen that  $\vee$  is generated by  $\{\neg, \wedge\}$  through  $P \vee Q \equiv \neg(\neg P \wedge \neg Q)$ , this will prove that  $\mathcal{C}_0 = \{\neg, \wedge\}$  is a complete set of connectives.

We will represent each abstract function  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  by an explicit formula called a **disjunctive normal form**.

#### 3.1. Step 1

We begin by representing some very simple functions.

Fix  $a = (\varepsilon_1, \dots, \varepsilon_n) \in \{0, 1\}^n$ . Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be the function such that  $f(a) = 1$  and  $f(x) = 0$  if  $x \neq a$ .

The following formula  $\mathcal{P}_a$  represents this function:

$$\mathcal{P}_a : \quad \varepsilon_1 P_1 \wedge \varepsilon_2 P_2 \wedge \dots \wedge \varepsilon_n P_n$$

Here,  $P_i$  denotes the  $i$ th propositional variable, that is, the  $i$ th coordinate of  $\{0, 1\}^n$ . Recall that  $\varepsilon_i = 0$  or  $\varepsilon_i = 1$ , and define

$$\varepsilon_i P_i = \begin{cases} P_i & \text{if } \varepsilon_i = 1 \\ \neg P_i & \text{if } \varepsilon_i = 0 \end{cases}$$

We will use the following notation:

$$\mathcal{P}_a : \quad \bigwedge_{i=1}^n \varepsilon_i P_i$$

This notation is analogous to  $\sum_{i=1}^n x_i$  for  $x_1 + x_2 + \dots + x_n$ , or to  $\prod_{i=1}^n x_i$  for  $x_1 x_2 \dots x_n$ .

For example, the formula  $P \wedge \neg Q \wedge \neg R$  gives a function that equals 1 at  $(1, 0, 0)$  and equals 0 everywhere else.

Recall that  $a = (\varepsilon_1, \dots, \varepsilon_n)$ . Let us prove that  $\mathcal{P}_a$  represents the function described above:

$$\begin{aligned} \mathcal{P}_a \text{ is true} & \quad \text{iff} \quad \varepsilon_i P_i \text{ is true, for every } i = 1, \dots, n \\ & \quad \text{iff} \quad \begin{cases} P_i \text{ is true} & \text{if } \varepsilon_i = 1 \\ P_i \text{ is false} & \text{if } \varepsilon_i = 0 \end{cases} \quad \text{for every } i = 1, \dots, n \end{aligned}$$

Therefore the associated function  $f_{\mathcal{P}_a}$  satisfies  $f_{\mathcal{P}_a}(x_1, \dots, x_n) = 1$  if and only if  $x_i = \varepsilon_i$  for every  $i = 1, \dots, n$ . In other words,  $f_{\mathcal{P}_a}(x) = 1$  if and only if  $x = a$ .

#### 3.2. Step 2

Let  $a_1, a_2, \dots, a_k \in \{0, 1\}^n$  be a list of  $k$  elements of  $\{0, 1\}^n$  (with  $k \geq 1$ ).

Let  $\mathcal{P}_{a_1, \dots, a_k}$  be the following formula:

$$\mathcal{P}_{a_1, \dots, a_k} : \quad \mathcal{P}_{a_1} \vee \mathcal{P}_{a_2} \vee \dots \vee \mathcal{P}_{a_k}$$

In other words:

$$\mathcal{P}_{a_1, \dots, a_k} : \quad \bigvee_{j=1}^k \mathcal{P}_{a_j} = \bigvee_{j=1}^k \bigwedge_{i=1}^n \varepsilon_{i,j} P_i$$

where we have written  $a_j = (\varepsilon_{1,j}, \varepsilon_{2,j}, \dots, \varepsilon_{n,j})$ .

Let  $f_{a_1, \dots, a_k}$  denote the function associated with the formula  $\mathcal{P}_{a_1, \dots, a_k}$ . Then:

$$f_{a_1, \dots, a_k}(x) = 1 \quad \text{iff} \quad x \in \{a_1, a_2, \dots, a_k\}$$

Example. For the formula  $\mathcal{P}_a \vee \mathcal{P}_b$ , the associated function  $f_{a,b}$  satisfies  $f(a) = 1$ ,  $f(b) = 1$ , and  $f(x) = 0$  for every other  $x$ .

Let us prove our claim.

- If  $x \in \{a_1, a_2, \dots, a_k\}$ , then one of the  $\mathcal{P}_{a_j}$  is true, so  $\mathcal{P}_{a_1, \dots, a_k}$  is true. Thus  $f_{a_1, \dots, a_k}(x) = 1$ .
- If  $x \notin \{a_1, a_2, \dots, a_k\}$ , then all the  $\mathcal{P}_{a_1}, \dots, \mathcal{P}_{a_k}$  are false, so  $\mathcal{P}_{a_1, \dots, a_k}$  is false. Thus  $f_{a_1, \dots, a_k}(x) = 0$ .

### 3.3. Step 3

Let  $f : \{0, 1\}^n \rightarrow \{0, 1\}$  be any abstract function.

- If  $f(x) = 0$  for every  $x \in \{0, 1\}^n$ , set  $\mathcal{P} = P_1 \wedge \neg P_1$  (always false); then  $f_{\mathcal{P}} = f$ .
- Otherwise, let  $\{a_1, \dots, a_k\}$  be the elements of  $\{0, 1\}^n$  for which  $f(x) = 1$ . Then  $f = f_{a_1, \dots, a_k}$  by Step 2.

This completes the proof.

### 3.4. Remarks

- The decomposition

$$\mathcal{P} : \bigvee_{j=1}^k \bigwedge_{i=1}^n \varepsilon_{i,j} P_i$$

is called the *disjunctive normal form*.

This normal form is useful because the decomposition is unique up to order (of course,  $P \vee Q \equiv Q \vee P$  and  $P \wedge Q \equiv Q \wedge P$ ).

- A variant is the *conjunctive normal form*:

$$\mathcal{Q} : \bigwedge_{j=1}^k \bigvee_{i=1}^n \delta_{i,j} P_i$$

where  $\delta_{i,j} \in \{0, 1\}$ . Its expression is also unique (up to order).

#### Example.

Let  $\mathcal{P}$  be the formula  $P \leftrightarrow Q$ . Its disjunctive normal form is:

$$\mathcal{P} \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$$

Its conjunctive normal form is:

$$\mathcal{P} \equiv (\neg P \vee Q) \wedge (P \vee \neg Q)$$

## Exercises

### Evaluating formulas

**Exercise 1.** Find a formula for  $P \oplus Q$  (exclusive or) involving only the connectives  $\neg$ ,  $\wedge$ , and  $\vee$ . Then find a formula involving only  $\neg$  and  $\leftrightarrow$ .

**Exercise 2.**

1. For  $n = 3$ , how many functions  $f : \{0, 1\}^3 \rightarrow \{0, 1\}$  are there?
2. The “majority” function returns **1** if at least two of the variables  $P, Q, R$  equal **1**. Find a formula for this function involving only the connectives  $\neg$ ,  $\wedge$ , and  $\vee$ .
3. When is  $P \oplus Q \oplus R$  true? What is its relationship with  $P \leftrightarrow (Q \leftrightarrow R)$ ?
4. The “exactly one” function returns **1** when exactly one of the variables  $P, Q$ , or  $R$  equals **1**. Find a formula for this function involving only the connectives  $\neg$ ,  $\wedge$ , and  $\vee$ .

**Exercise 3.** Let  $n \geq 2$ .

1. Find a formula that returns true unless all the variables  $P_1, \dots, P_n$  are true.
2. What formula returns true when at least two variables are true?

## A complete set of connectives

**Exercise 4.**

1. Show that  $\mathcal{C} = \{\neg, \leftrightarrow\}$  is a complete set of connectives.
2. Show that  $\mathcal{C} = \{\vee, \leftrightarrow, \perp\}$  is a complete set of connectives. Recall that  $\perp$  is the always-false formula.
3. Show that  $\mathcal{C} = \{\wedge, \vee\}$  is not a complete set of connectives.
4. Show that  $\mathcal{C} = \{\wedge, \oplus\}$  is not a complete set of connectives ( $\oplus$  denotes exclusive or).

**Exercise 5.** Show that  $\{\downarrow\}$  forms a complete set. Recall that  $P \downarrow Q \equiv \neg(P \vee Q)$ .

## Proof

**Exercise 6.** Let  $n = 2$ . Only the connectives  $\neg, \wedge, \vee$  are allowed.

1. Find a formula that represents the function  $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ , which equals 1 at  $(1, 0)$  and equals 0 everywhere else.
2. Answer the same question for  $g : \{0, 1\}^2 \rightarrow \{0, 1\}$ , which equals 1 at  $(1, 1)$  and 0 everywhere else.
3. Deduce a formula for  $h : \{0, 1\}^2 \rightarrow \{0, 1\}$ , which equals 1 exactly at  $(1, 0)$  and  $(1, 1)$ . Simplify this formula.

**Exercise 7.** Let  $n = 2$ .

1. What function is associated with the formula  $P \rightarrow Q$ ?
2. Find the associated disjunctive normal form.

**Exercise 8.** Let  $n = 3$ . Only the connectives  $\neg, \wedge, \vee$  are allowed.

1. Find a formula that represents the function  $f$ , which equals 1 only at  $(1, 0, 1)$ .
2. Find a formula that represents the function  $g$ , which equals 0 only at  $(0, 1, 1)$ .
3. Find a formula that represents the function  $h$ , which equals 1 only at  $(1, 0, 0)$ ,  $(0, 1, 0)$ ,  $(0, 0, 1)$ . Find a disjunctive normal form and a conjunctive normal form for it.

# Compactness Theorem

*The Compactness Theorem states that when there are infinitely many assumptions, it is sufficient to use finitely many of them.*

*This chapter is more difficult than the preceding ones and may be skipped on a first reading.*

## 1. Formulas, satisfiable formulas, logical consequence

### 1.1. Formulas: definition from above

In the chapter “True/False Logic”, we defined a formula as a finite word constructed from the following elements:

- propositional variables  $\text{Var} = \{P_i\}_{i \in I}$ ,
- parentheses ( and ),
- and the logical connectives  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ .

More precisely:

#### Definition.

The set  $\text{Form}$  of **formulas** is constructed according to the following rules:

1. Propositional variables are formulas.
2. If  $\mathcal{P}$  and  $\mathcal{Q}$  are formulas, then

$$\neg \mathcal{P}, \quad (\mathcal{P} \wedge \mathcal{Q}), \quad (\mathcal{P} \vee \mathcal{Q}), \quad (\mathcal{P} \rightarrow \mathcal{Q}), \quad (\mathcal{P} \leftrightarrow \mathcal{Q})$$

are also formulas.

3. Nothing else is a formula.

This is called the “definition from above”, or the “inductive definition”, of formulas. We saw in the chapter “Which Connectives Are Useful?” that the connectives  $\{\neg, \wedge\}$  alone are sufficient to describe every formula. In this chapter, we are interested in the case where the set  $\text{Var} = \{P_i\}_{i \in I}$  of propositional variables is infinite but countable. This means that we may take  $I = \mathbb{N}$  as the index set for these variables:  $P_0, P_1, P_2, \dots$

#### Remark.

Since there are infinitely many variables, infinitely many indices  $i$  would be needed to describe all the formulas. But we want a formula to be a word formed from a finite alphabet. To achieve this, we will use only two symbols to describe all the variables:  $P$  and  $'$  (prime). Thus we write  $P$  for  $P_0$ ,  $P'$  for  $P_1$ ,  $P''$  for  $P_2$ , and so on. A formula is therefore a finite word composed from the finite alphabet  $\{P, ', (, ), \neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ . Of course, in practice we will retain our notation  $P_i$  for clarity.

## 1.2. Formulas: definition from below

The preceding definition is actually rather abstract and does not describe the set of formulas very well. Here is a new definition, called the “definition from below” or the “recursive definition”.

### Definition.

We define sets of formulas  $\text{Form}_n$  recursively.

- $\text{Form}_0 = \{P_i\}_{i \in I}$  is the set of variables.
- For  $n \geq 0$ ,

$$\text{Form}_{n+1} = \text{Form}_n \cup \{\neg \mathcal{P} \mid \mathcal{P} \in \text{Form}_n\} \cup \{(\mathcal{P} \wedge \mathcal{Q}) \mid \mathcal{P}, \mathcal{Q} \in \text{Form}_n\} \cup \dots$$

(Taking the corresponding union for all the connectives  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ .)

- $\text{Form} = \bigcup_{n \geq 0} \text{Form}_n$  is the union of all the  $\text{Form}_n$ .
- The **height** of a formula  $\mathcal{P}$  is the least integer  $n$  such that  $\mathcal{P} \in \text{Form}_n$ .

### Proposition 1.

If the set  $\text{Var} = \{P_i\}_{i \in I}$  of propositional variables is countable (or finite), then the set  $\text{Form}$  of formulas is also countable.

*Proof.* The set  $\text{Form} = \bigcup_{n \geq 0} \text{Form}_n$  is a union indexed by a countable set, in which each set  $\text{Form}_n$  is countable, so  $\text{Form}$  is countable.  $\square$

Throughout the rest of this chapter, we will assume that the set of propositional variables is finite or countable, and hence that the set of all formulas is countable.

## 1.3. Evaluating a formula

We have already seen that evaluating a formula  $\mathcal{P}$  consists of assigning a value True or False to each propositional variable  $P_i$ , in order to assign a T/F value to the formula  $\mathcal{P}$  being evaluated.

Here is the rigorous definition of an evaluation based on the definition of formulas “from above”. Here **1** and **0** denote the two Boolean values T/F.

### Definition.

- Let  $v : \{P_i\}_{i \in I} \rightarrow \{0, 1\}$  be a function, called an **assignment of truth values** or a **valuation**: for every  $i \in I$ , we have  $v(P_i) = 0$  or  $v(P_i) = 1$ .
- We extend this function to a function  $v : \text{Form} \rightarrow \{0, 1\}$  on the set  $\text{Form}$  of formulas:
  - $v(\neg \mathcal{P}) = 1$  iff  $v(\mathcal{P}) = 0$ ,
  - $v(\mathcal{P} \wedge \mathcal{Q}) = 1$  iff  $v(\mathcal{P}) = 1$  and  $v(\mathcal{Q}) = 1$ ,
  - and according to the same principle for  $\vee, \rightarrow, \leftrightarrow$ .
- The value  $v(\mathcal{P})$  is the **evaluation** of  $\mathcal{P}$  under  $v$ .

## 1.4. Satisfiability, logical consequence

### Definition.

- A formula  $\mathcal{P}$  is **satisfiable** if there exists a valuation  $v$  such that  $v(\mathcal{P})$  is true. We then say that  $v$  **satisfies**  $\mathcal{P}$ , or that  $v$  is a **model** for  $\mathcal{P}$ .
- A set of formulas  $\Gamma = \{\mathcal{P}_j\}_{j \in J}$  is **satisfiable** if there exists a valuation  $v$  such that, for every  $j \in J$ ,  $v(\mathcal{P}_j)$  is true.
- A formula, or a set of formulas, that is not satisfiable is said to be **unsatisfiable** or **contradictory**.

**Definition.**

Let  $\Gamma$  be a set of formulas and  $\mathcal{Q}$  a formula. We say that  $\mathcal{Q}$  is a *logical consequence* of  $\Gamma$  if every valuation  $v$  that satisfies  $\Gamma$  also satisfies  $\mathcal{Q}$ . We write:

$$\Gamma \models \mathcal{Q}$$

This means that  $\Gamma \cup \{\neg \mathcal{Q}\}$  is not satisfiable; that is, no assignment of truth values can satisfy both  $\Gamma$  and  $\neg \mathcal{Q}$ .

We should think of  $\Gamma$  as assumptions and  $\mathcal{Q}$  as a conclusion. Whenever all the assumptions hold, the conclusion does as well.

## 2. Compactness Theorem

### 2.1. Statement

Let  $\Gamma$  be a finite or infinite set of formulas.

**Theorem 1.**

$\Gamma$  is satisfiable if and only if, for every finite subset  $\Gamma_0 \subset \Gamma$ ,  $\Gamma_0$  is satisfiable.

To make the proof easier to write, we will say that  $\Gamma$  is *finitely satisfiable* if, for every finite subset  $\Gamma_0 \subset \Gamma$ ,  $\Gamma_0$  is satisfiable. Thus the Compactness Theorem states:

$$\Gamma \text{ satisfiable} \quad \text{iff} \quad \Gamma \text{ finitely satisfiable}$$

Remarks:

1. The forward implication is obvious. If  $\Gamma$  is satisfiable, then every subset of  $\Gamma$  is also satisfiable, so in particular  $\Gamma$  is finitely satisfiable.
2. If  $\Gamma$  is a finite set, then the theorem is obvious; there is nothing to prove.
3. If  $\Gamma$  is infinite, the converse is far from obvious. Some results that hold for finite sets do not necessarily hold for infinite sets. For example, a finite intersection of open intervals is an open interval (possibly empty). However, this is no longer true for an infinite intersection:

$$\bigcap_{n \geq 1} \left] -\frac{1}{n}, 1 + \frac{1}{n} \right[ = [0, 1]$$

This is a striking example in which an infinite intersection of open intervals results in a closed interval.

### 2.2. Corollaries

The following corollaries are also called the “Compactness Theorem”.

**Corollary 1.**

$\Gamma$  is unsatisfiable if and only if there exists a finite set  $\Gamma_0 \subset \Gamma$  such that  $\Gamma_0$  is unsatisfiable.

In other words, if a set of formulas cannot be satisfied, this cannot be due to the fact that it contains infinitely many formulas.

Here is the positive counterpart:

**Corollary 2.**

$\Gamma \models \mathcal{Q}$  if and only if there exists a finite set  $\Gamma_0 \subset \Gamma$  such that  $\Gamma_0 \models \mathcal{Q}$ .

This is the version that interests us most: if we have infinitely many formulas  $\Gamma$  as assumptions, with the formula  $\mathcal{Q}$  as a logical consequence, then we can extract a finite subset  $\Gamma_0$  that still entails the same logical consequence  $\mathcal{Q}$ . Thus we go from infinitely many assumptions to finitely many.

## 2.3. Proof of the corollaries

Without waiting for the proof of the theorem, here is a proof of the corollaries.

*Proof of Corollary 1.* The reverse implication  $\Leftarrow$  is clear: if  $\Gamma_0$  is not satisfiable, then no set  $\Gamma$  containing  $\Gamma_0$  will be satisfiable.

The forward implication  $\Rightarrow$  is proved using the Compactness Theorem, which states:

$$\forall \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ satisfiable} \quad \rightarrow \quad \Gamma \text{ satisfiable}$$

(Only finite subsets  $\Gamma_0$  are considered.) Its contrapositive is therefore:

$$\Gamma \text{ not satisfiable} \quad \rightarrow \quad \text{not } (\forall \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ satisfiable})$$

This can be written:

$$\Gamma \text{ unsatisfiable} \quad \rightarrow \quad \exists \Gamma_0 \subset \Gamma \quad \Gamma_0 \text{ unsatisfiable}$$

□

*Proof of Corollary 2.* We will use the fact that “ $\Gamma \models Q$  if and only if  $\Gamma \cup \{\neg Q\}$  is unsatisfiable”.

The reverse implication  $\Leftarrow$  is the easy one: let  $\Gamma_0$  be finite with  $\Gamma_0 \models Q$ . Then  $\Gamma_0 \cup \{\neg Q\}$  is unsatisfiable.

For every set  $\Gamma$  containing  $\Gamma_0$ , it follows a fortiori that  $\Gamma \cup \{\neg Q\}$  is unsatisfiable. Thus  $\Gamma \models Q$ .

Let us prove the forward implication  $\Rightarrow$  using Corollary 1. Suppose  $\Gamma \models Q$ . Then  $\Gamma \cup \{\neg Q\}$  is unsatisfiable. By Corollary 1, there exists a finite set  $\Delta_0 \subset \Gamma \cup \{\neg Q\}$  such that  $\Delta_0$  is unsatisfiable.

- If  $\neg Q \in \Delta_0$ , then we write  $\Delta_0 = \Gamma_0 \cup \{\neg Q\}$  with  $\Gamma_0 \subset \Gamma$  finite. Thus  $\Gamma_0 \cup \{\neg Q\}$  is unsatisfiable, which is equivalent to  $\Gamma_0 \models Q$ .
- If  $\neg Q \notin \Delta_0$ , then we simply set  $\Gamma_0 = \Delta_0$  which is unsatisfiable. A fortiori,  $\Gamma_0 \cup \{\neg Q\}$  is unsatisfiable, so  $\Gamma_0 \models Q$ .

In both cases, we have proved  $\Gamma_0 \models Q$ .

□

## 3. Proof

The forward implication  $\Rightarrow$  of the Compactness Theorem is clear: indeed, if  $\Gamma$  is satisfiable, then every subset  $\Gamma_0 \subset \Gamma$  (finite or not) is satisfiable. We therefore focus on proving the converse: if  $\Gamma$  is finitely satisfiable, then  $\Gamma$  is satisfiable.

### 3.1. Part 1

Let  $\Gamma$  be a finitely satisfiable set of formulas. First we will construct a set  $\Delta$  such that:

1.  $\Gamma \subset \Delta$ ,
2. for every formula  $Q$  in the set Form of all formulas, either  $Q \in \Delta$  or  $\neg Q \in \Delta$ ,
3.  $\Delta$  is finitely satisfiable.

We will recursively construct increasingly large finitely satisfiable sets  $\Delta_n$ . The set  $\Delta$  will be the union of all the  $\Delta_n$ . Let us begin by defining:

$$\Delta_0 = \Gamma$$

By assumption,  $\Delta_0$  is finitely satisfiable.

We assumed that the set Var of propositional variables was countable, so by Proposition 1, the set Form of all formulas is also countable. We can therefore enumerate it using integers:

$$\text{Form} = \{Q_1, Q_2, \dots\}$$

Let us first show how to construct  $\Delta_1$ . We distinguish two cases:

- If  $\Delta_0 \cup \{Q_1\}$  is finitely satisfiable, we simply set:

$$\Delta_1 = \Delta_0 \cup \{Q_1\}$$

- Otherwise, we set:

$$\Delta_1 = \Delta_0 \cup \{\neg Q_1\}$$

But in this case too,  $\Delta_1$  is finitely satisfiable, by the lemma below.

Thus in every case,  $\Delta_1$  is finitely satisfiable.

**Lemma 1.**

*Let  $\Sigma$  be a finitely satisfiable set and let  $Q$  be any formula. Then  $\Sigma \cup \{Q\}$  or  $\Sigma \cup \{\neg Q\}$  is finitely satisfiable.*

*Proof.* Suppose for contradiction that  $\Sigma \cup \{Q\}$  is not finitely satisfiable and that  $\Sigma \cup \{\neg Q\}$  is not finitely satisfiable either. Thus:

- there exists a finite set  $\Sigma_1 \cup \{Q\}$  that is unsatisfiable (with  $\Sigma_1 \subset \Sigma$ ),
- and there exists a finite set  $\Sigma_2 \cup \{\neg Q\}$  that is unsatisfiable (with  $\Sigma_2 \subset \Sigma$ ).

Let  $\Sigma_0 = \Sigma_1 \cup \Sigma_2$ . It is a finite set contained in  $\Sigma$ . By assumption, it is therefore satisfiable. Let  $v$  be a valuation that satisfies  $\Sigma_0$ . If  $v$  satisfies  $Q$ , then  $\Sigma_1 \cup \{Q\}$  is satisfied, which is impossible. But if  $v$  does not satisfy  $Q$ , then  $v$  satisfies  $\neg Q$  and hence  $v$  satisfies  $\Sigma_2 \cup \{\neg Q\}$ , which is also impossible. We have therefore obtained a contradiction.  $\square$

Let us continue our recursive construction. Suppose that we have already constructed a finitely satisfiable  $\Delta_n$  of the form:

$$\Delta_n = \Gamma \cup \{\varepsilon_1 Q_1, \dots, \varepsilon_n Q_n\}$$

where  $\varepsilon_i Q_i$  denotes either  $Q_i$  or  $\neg Q_i$ . We then define:

$$\Delta_{n+1} = \begin{cases} \Delta_n \cup \{Q_{n+1}\} & \text{if it is finitely satisfiable,} \\ \Delta_n \cup \{\neg Q_{n+1}\} & \text{otherwise.} \end{cases}$$

By Lemma 1,  $\Delta_{n+1}$  is finitely satisfiable.

Now define:

$$\Delta = \bigcup_{n \geq 0} \Delta_n$$

Since  $\Delta_0 \subset \Delta_1 \subset \Delta_2 \subset \dots$ , we should think of  $\Delta$  as the limit of the  $\Delta_n$  as  $n$  tends to  $+\infty$ . Let us now verify the stated properties:

1.  $\Gamma = \Delta_0 \subset \Delta$ ,
2. for every formula  $Q_n$ , either  $Q_n \in \Delta_n \subset \Delta$  or  $\neg Q_n \in \Delta_n \subset \Delta$ ,
3. finally,  $\Delta$  is finitely satisfiable. Indeed, let  $\Gamma_0 \subset \Delta$  be a finite set. Then there exists  $n \geq 0$  such that  $\Gamma_0 \subset \Delta_n$ . But then  $\Gamma_0$  is satisfiable because  $\Delta_n$  is finitely satisfiable.

### 3.2. Part 2

We will now prove that all the formulas  $\mathcal{P}$  in  $\Delta$  are simultaneously satisfiable. We will even specify the valuation  $v$  that satisfies them. Since  $\Gamma \subset \Delta$ , it will follow that  $\Gamma$  is satisfiable.

Let us define this valuation  $v$ . Write  $\text{Var} = \{P_i\}_{i \in I}$  for the set of propositional variables. Let  $v$  be the valuation defined by:

$$\begin{cases} v(P_i) = \mathbf{1} & \text{if } P_i \in \Delta, \\ v(P_i) = \mathbf{0} & \text{otherwise.} \end{cases}$$

Once defined on the propositional variables,  $v$  extends to all formulas (see Section 1.3).

Let  $\mathcal{P}$  be any formula. We will prove:

**Lemma 2.**

$\mathcal{P}$  is satisfied by  $\nu$  if and only if  $\mathcal{P} \in \Delta$ .

Thus, since  $\Gamma \subset \Delta$ , every formula  $\mathcal{P} \in \Gamma$  is satisfied by  $\nu$ , and so  $\Gamma$  is satisfiable because it is satisfied by  $\nu$ . It remains to prove the lemma.

*Proof.* The proof is by induction on the height of  $\mathcal{P}$ . To simplify the proof, suppose that the only connectives used are  $\neg$  and  $\wedge$ . This is possible without loss of generality by the chapter “Which Connectives Are Useful?”

- If  $\mathcal{P}$  is one of the propositional variables  $P_i$ , then by definition of  $\nu$ ,  $\nu$  satisfies  $\mathcal{P}$  if and only if  $\mathcal{P} \in \Delta$ .
- Suppose that the assertion of the lemma holds for the formula  $\mathcal{P}$ ; let us prove that it also holds for  $\neg\mathcal{P}$ .

$$\begin{aligned}
 (\neg\mathcal{P}) \text{ satisfied by } \nu &\iff \nu(\neg\mathcal{P}) = 1 \\
 &\iff \nu(\mathcal{P}) = 0 \\
 &\iff \text{not}(\mathcal{P} \text{ satisfied by } \nu) \\
 &\iff \text{not}(\mathcal{P} \in \Delta) \\
 &\iff \mathcal{P} \notin \Delta \\
 &\iff \neg\mathcal{P} \in \Delta
 \end{aligned}$$

For the last line, recall that property (2) of  $\Delta$  tells us that either  $\mathcal{P}$  or  $\neg\mathcal{P}$  belongs to  $\Delta$ .

- Suppose that the assertion of the lemma holds for the formulas  $\mathcal{P}$  and  $\mathcal{Q}$ ; let us prove that it also holds for  $\mathcal{P} \wedge \mathcal{Q}$ .

Forward implication  $\Rightarrow$ . If  $\mathcal{P} \wedge \mathcal{Q}$  is satisfied by  $\nu$ , let us show that  $\mathcal{P} \wedge \mathcal{Q} \in \Delta$ .

Since  $\nu$  satisfies  $\mathcal{P} \wedge \mathcal{Q}$ ,  $\nu$  satisfies  $\mathcal{P}$ , so  $\mathcal{P} \in \Delta$ , and  $\nu$  satisfies  $\mathcal{Q}$ , so  $\mathcal{Q} \in \Delta$ .

Suppose for contradiction that  $\mathcal{P} \wedge \mathcal{Q} \notin \Delta$ . Then, by property (2) of  $\Delta$ , we have  $\neg(\mathcal{P} \wedge \mathcal{Q}) \in \Delta$ . Now consider the set  $\{\mathcal{P}, \mathcal{Q}, \neg(\mathcal{P} \wedge \mathcal{Q})\}$ . This is a finite, unsatisfiable set of formulas (no valuation can simultaneously satisfy  $\mathcal{P}$ ,  $\mathcal{Q}$ , and  $\neg(\mathcal{P} \wedge \mathcal{Q})$ ). This contradicts the fact that  $\Delta$  is finitely satisfiable.

Conclusion:  $\mathcal{P} \wedge \mathcal{Q} \in \Delta$ .

Reverse implication  $\Leftarrow$ . If  $\mathcal{P} \wedge \mathcal{Q} \in \Delta$ , let us show that  $\mathcal{P} \wedge \mathcal{Q}$  is satisfied by  $\nu$ .

Suppose for contradiction that  $\mathcal{P} \notin \Delta$ . Then  $\neg\mathcal{P} \in \Delta$  (again by property (2) of  $\Delta$ ). Thus  $\{\mathcal{P} \wedge \mathcal{Q}, \neg\mathcal{P}\} \subset \Delta$  is a finite set, but of course it is not satisfiable. This again contradicts the fact that  $\Delta$  is finitely satisfiable. Thus  $\mathcal{P} \in \Delta$ , so by the induction hypothesis  $\nu(\mathcal{P}) = 1$ .

Similarly, we prove that  $\mathcal{Q} \in \Delta$  and hence  $\nu(\mathcal{Q}) = 1$ . But then  $\nu(\mathcal{P} \wedge \mathcal{Q}) = 1$  (by the construction of the valuation), and so  $\mathcal{P} \wedge \mathcal{Q}$  is satisfied by  $\nu$ .

□

**Remark.** If we allow the other classical connectives  $\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$ , then we can prove in the same way that  $\mathcal{P} \vee \mathcal{Q}$  is satisfied by  $\nu$  if and only if  $\mathcal{P} \vee \mathcal{Q} \in \Delta$ , and so on. (This is left as an exercise.)

## 4. $\{0, 1\}^{\mathbb{N}}$ is compact

Why is the Compactness Theorem called by this name? Here we make the connection with the notion of a compact set. This section is intended for readers who already know topology, whose concepts will not be reviewed here.

### 4.1. Topology of $\{0, 1\}^{\mathbb{N}}$

- Let  $\mathbb{B} = \{0, 1\}$  have the discrete topology; that is,  $\{0\}$  and  $\{1\}$  are both open and closed sets.
- $\mathbb{B}^{\mathbb{N}} = \{0, 1\}^{\mathbb{N}}$  is a product space. It may be viewed as the set  $\{v : \mathbb{N} \rightarrow \{0, 1\}\}$  of functions that assign a value 0 or 1 to each integer  $i$ . In other words, it is the set of valuations that assign a T/F value to each propositional variable  $P_i$  (here  $\text{Var} = \{P_i\}_{i \in \mathbb{N}}$  is countable).
- We equip  $\{0, 1\}^{\mathbb{N}}$  with the product topology; by definition, this is the coarsest topology that makes the projections  $\pi_i : \{0, 1\}^{\mathbb{N}} \rightarrow \{0, 1\}$  continuous.
- Thus, for fixed  $i$ ,  $\pi_i^{-1}(\{1\}) = \{v : \mathbb{N} \rightarrow \{0, 1\} \mid v(i) = 1\}$  is closed because it is the inverse image of a closed set under a continuous map. It is also open because it is the inverse image of an open set, which makes its complement closed.
- Let  $\mathcal{P}$  be a formula. Write  $V(\mathcal{P}) = \{v \in \{0, 1\}^{\mathbb{N}} \mid v(\mathcal{P}) = 1\}$ . The set  $V(\mathcal{P})$  is closed. The proof is by induction on the height of  $\mathcal{P}$  (much as in the second part of the proof of the Compactness Theorem; see the exercises).
- A formula  $\mathcal{P}$  is satisfiable if and only if  $V(\mathcal{P})$  is nonempty.

### 4.2. A product of compact spaces is compact

**Theorem 2** (Tychonoff's theorem).

*An arbitrary product of compact spaces is compact.*

The product may be finite or infinite (countable or not).

**Corollary 3.**

$\{0, 1\}^{\mathbb{N}}$  is a compact space.

Indeed,  $\{0, 1\}$  is compact because it is a finite set.

Here is an important property of compact spaces.

**Proposition 2.**

*Let  $F_j$ ,  $j \in J$ , be closed subsets of a compact space  $E$ . If  $\bigcap_{j \in J} F_j = \emptyset$ , then there exists a finite set  $J_0 \subset J$  such that  $\bigcap_{j \in J_0} F_j = \emptyset$ .*

This is equivalent to the more familiar version for open sets: “Every open cover of a compact space has a finite subcover.”

### 4.3. Proof of the Compactness Theorem

Let  $\Gamma = \{\mathcal{P}_0, \mathcal{P}_1, \dots\}$  be a finitely satisfiable set of formulas.

Write  $F_j = V(\mathcal{P}_j)$ , which is a closed subset of  $\{0, 1\}^{\mathbb{N}}$ . Note that  $F_j$  is nonempty because  $\mathcal{P}_j$  is satisfiable (since  $\{\mathcal{P}_j\}$  is a finite subset of  $\Gamma$ ). More generally, if  $\Gamma_0 = \{\mathcal{P}_j\}_{j \in J_0}$  is a finite set of formulas contained in  $\Gamma$ , then there exists a valuation  $v$  that simultaneously satisfies all the  $\mathcal{P}_j$ . Therefore  $v \in F_j$  for every  $j \in J_0$ . In other words:

$$\bigcap_{j \in J_0} F_j \neq \emptyset$$

Thus every finite intersection of closed sets is nonempty. By the contrapositive of Proposition 2, we therefore have:

$$\bigcap_{j \in \mathbb{N}} F_j \neq \emptyset$$

This means that there exists  $v : \mathbb{N} \rightarrow \{0, 1\}$  such that  $v(P_j) = 1$  for every  $j \in \mathbb{N}$ . Thus  $v$  satisfies  $\Gamma$ . We have indeed proved that  $\Gamma$  is satisfiable.

## Exercises

### Formulas, satisfiable formulas, logical consequence

**Exercise 1.** Consider  $\text{Var} = \{P, Q, R\}$  and  $\text{Conn} = \{\neg, \wedge\}$ . List all formulas of height 0, then of height 1. How many formulas of height 2 are there? Answer the same questions with  $\text{Var} = \{P, Q\}$  and  $\text{Conn} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ .

**Exercise 2.** Show that two formulas are logically equivalent exactly when each is a logical consequence of the other.

$$\mathcal{P} \equiv \mathcal{Q} \quad \text{iff} \quad \mathcal{P} \models \mathcal{Q} \text{ and } \mathcal{Q} \models \mathcal{P}$$

**Exercise 3.** Find all valuations defined on  $\{P_i\}_{i \in \mathbb{N}}$  that satisfy the infinite set of formulas:

$$\Gamma = \{P_0 \rightarrow P_1, P_1 \rightarrow P_2, P_2 \rightarrow P_3, \dots\}$$

### Compactness Theorem

**Exercise 4.** For the variables  $\{P_i\}_{i \in \mathbb{N}}$ , define, for  $n \geq 0$ , the formula  $\mathcal{P}_n = \neg(P_n \wedge P_{n+1})$ . Let  $\Gamma = \{\mathcal{P}_n \mid n \geq 0\}$ .

1. Show that every finite subset of  $\Gamma$  is satisfiable.
2. Deduce that  $\Gamma$  is satisfiable.
3. Characterize all valuations defined on  $\{P_i\}_{i \in \mathbb{N}}$  that satisfy  $\Gamma$ .

**Exercise 5.** For each integer  $n \in \mathbb{N}$ , define the formula:

$$\mathcal{P}_n = (P_n \vee P_{n+1}) \wedge (\neg P_{n+2}).$$

Consider the set  $\Gamma = \{\mathcal{P}_n \mid n \geq 0\}$ .

1. For fixed  $n \geq 0$ , show that  $\{\mathcal{P}_n\}$  is satisfiable.
2. For fixed  $n \geq 0$ , show that  $\{\mathcal{P}_n, \mathcal{P}_{n+1}\}$  is satisfiable.
3. Is  $\Gamma$  satisfiable?

**Exercise 6.** Starting from Corollary 2, which may be assumed, give a proof of the Compactness Theorem.

## Proof

**Exercise 7.** Extend the proof of Lemma 2 to show directly that:

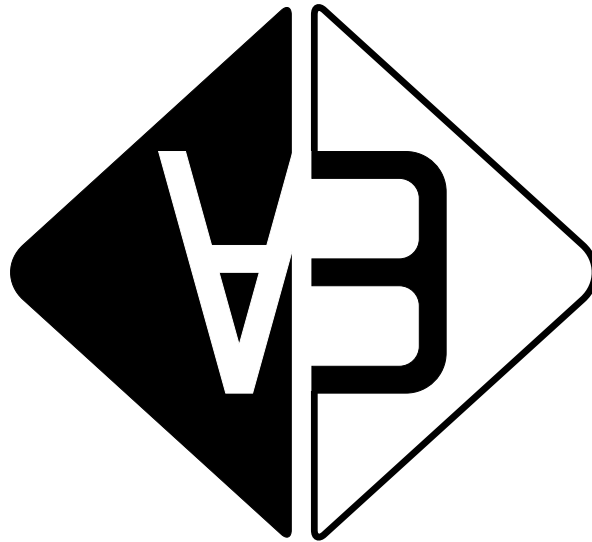
$$v \text{ satisfies } \mathcal{P} \vee \mathcal{Q} \quad \text{iff} \quad \mathcal{P} \vee \mathcal{Q} \in \Delta$$

**$\{0, 1\}^{\mathbb{N}}$  is compact**

**Exercise 8.** Let  $\mathcal{P}$  be a formula and  $V(\mathcal{P}) = \{v \in \{0, 1\}^{\mathbb{N}} \mid v(\mathcal{P}) = 1\}$ . Show that  $V(\mathcal{P})$  is a closed subset of  $\{0, 1\}^{\mathbb{N}}$ . First justify this for propositional variables, then prove it by induction on the height of  $\mathcal{P}$ , using the proof of Lemma 2 as a guide.

---

## PART II



## FIRST-ORDER LOGIC

---

# First-Order Logic

We consider a language with variables and the quantifiers “for all” and “there exists.” We enrich this language with predicates and functions.

## 1. Motivation

### 1.1. Quantifiers

The true/false logic of the previous chapters is ultimately rather limited and cannot express the natural statements found in a mathematics course. In this chapter, we consider a more powerful language: first-order logic. It allows us to write the following statements:

- “ $\forall n \in \mathbb{Z} \quad n^2 \geq 0$ ”: “the square of every integer is nonnegative.”
- “ $\exists z \in \mathbb{C} \quad z^2 = -1$ ”: “there exists a complex number whose square is  $-1$ .”
- “ $\forall g \in G \quad \exists g' \in G \quad g \circ g' = e$ ”: “every element of  $G$  has an inverse,” for a group  $(G, \circ)$  with identity element  $e$ .

With this language, we can define complicated notions with subtle nuances. For example, let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be a function.

- Continuity on  $\mathbb{R}$  is defined by:

$$\forall x \in \mathbb{R} \quad \forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall y \in \mathbb{R} \quad |x - y| < \delta \rightarrow |f(x) - f(y)| < \varepsilon$$

- Whereas uniform continuity is defined by:

$$\forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad |x - y| < \delta \rightarrow |f(x) - f(y)| < \varepsilon$$

### 1.2. New features

The main new feature is, of course, the appearance of the quantifiers “ $\forall$ ” and “ $\exists$ .” The single statement “ $\forall n \in \mathbb{N} \quad n^2 \geq 0$ ” expresses infinitely many truths. In true/false logic, we would not know how to prove such a claim, because we would have to consider the case  $n = 0$ , then  $n = 1$ , and so on, requiring infinitely many steps. Here we treat the statement “ $\forall n \in \mathbb{N} \quad n^2 \geq 0$ ” as a single claim to be proved.

The second advantage of first-order logic is that it applies to a wide variety of situations. In propositional logic, we considered formulas, and satisfaction consisted in replacing the variables with the values T/F. Here we still want to decide whether a formula is true or false, but the variables correspond to more general objects. For example, consider the formula:

$$(\forall x \quad P(x)) \vee (\exists x \quad \neg P(x))$$

It applies to  $x$  in any set, and also to any predicate  $P(x)$ . In words, it means “a property is true for every element, or there exists an element for which it is not true.” Here is what this formula becomes in three different structures:

- $(\forall x \in \mathbb{R} \quad x^2 \neq 0) \vee (\exists x \in \mathbb{R} \quad x^2 = 0)$
- $(\forall n \in \mathbb{N} \quad n + 1 > n) \vee (\exists n \in \mathbb{N} \quad n + 1 \leq n)$
- $(\forall g \in G \quad g \circ g = e) \vee (\exists g \in G \quad \neg(g \circ g = e))$ , in a group  $G$ .

First-order languages adapt to a wide variety of structures (sets, groups, fields, vector spaces...) and to the corresponding proofs.

### 1.3. Limitations

First-order logic is powerful; however, some things cannot be expressed in these languages. The most striking limitation is that there is no formula expressing whether a set is finite (which means, for example, that we cannot define the dimension of a vector space). Working in first-order logic means that we consider elements in a set  $E$ , written “ $x \in E$ ,” but we cannot quantify over subsets  $A \subset E$ . Thus, the statement “every nonempty subset has a least element” (which is the definition of a *well-ordered* set) is expressed in *second-order logic*, because it concerns subsets rather than elements. In conclusion, first-order logic describes local properties of objects well, but it is less effective at describing global properties related to infinity.

## 2. Language

A first-order language should be understood as a common general framework that we will then specialize to each situation (integers, real numbers, groups...).

### 2.1. Elements of the language

The formulas of first-order logic are constructed from languages made up of the following elements.

#### Logical symbols

- **variables**:  $x, y, z, x_1, x_2, \dots$
- parentheses: ( and )
- connectives:  $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- **quantifiers**:  $\forall$  (for all),  $\exists$  (there exists)

#### Nonlogical symbols

- **constants**:  $a, b, c, c_1, c_2, \dots$
- **predicates**:  $P(x), P(x, y), \dots, P(x_1, \dots, x_n)$ ; including equality  $x = y$
- **functions**:  $f(x), f(x, y), \dots, f(x_1, \dots, x_n)$

We discuss the new notions in detail below. We will first consider languages without functions; we will add them at the end of this chapter.

## 2.2. Logical symbols

The language is much richer than that of true/false logic. Of course, we again find the logical connectives (and parentheses), which will play the same role here as in propositional logic. Let us examine the new features in detail.

- **Universal quantifier**  $\forall$ . It is an upside-down A, introduced by Gerhard Gentzen in 1935 as the initial letter of *all* or *alle*. It is read “for all” or “for every.”
- **Existential quantifier**  $\exists$ . It is a backward E, introduced by Giuseppe Peano at the end of the nineteenth century. It is read “there exists” or “there is at least one.”
- **Variables.** They are used to discuss the objects under consideration without naming them individually. Thus, “ $\forall x P(x)$ ” or “ $\exists x P(x)$ ” applies to every  $x$  (or to some  $x$ ), but there is not really an object named  $x$ . This is called a **bound variable** (or a *dummy variable*). Thus, “ $\forall y P(y)$ ” would have the same meaning.

Note: for variables, we reserve the letters at the end of the alphabet,  $x, y, z$ , with subscripts if needed.

## 2.3. Nonlogical symbols

This is another new feature of first-order logic. We want constructions and theories that apply to many situations. Constants, predicates, and functions are like empty shells whose meaning must be specified in each application.

- **Constants.** Written  $a, b, c, c_1, c_2, \dots$ , these are symbols of the language that will distinguish certain elements. For example, we can define a language suited to groups with a constant  $c$  that will play the role of the identity element  $e, 0, 1, \dots$  depending on the group under consideration. When defining the language, we choose the names and number of constants.
- **Predicates.** A predicate is somewhat like a super-proposition in true/false logic: it returns a T/F value, but that value depends on the values of one or more parameters. Their symbols in the language will often be  $P, Q$ , or  $R$ .
  - Unary predicate  $P$ , written  $P(x)$ : “does  $x$  have property  $P$ ?” (this may be true or false depending on the value assigned to  $x$ ).  
For example, such a predicate could mean “ $x^2 \geq x$ ,” “ $x^2 = 1$ ,” or “ $x^5 + x^3 + 1 = 0$ ” ... It may be true or false depending on  $x$ .
  - Binary predicate  $P$ , written  $P(x, y)$ : “does  $(x, y)$  have property  $P$ ?” This can also be understood as “is  $x$  related to  $y$  by  $P$ ?”  
For example, a binary predicate could mean “ $x < y$ ” or “ $x^2 + y^2 = 1$ ”...
  - **Equality** “ $x = y$ ” is a special binary predicate. Unless otherwise stated (which will be rare), we will always assume that equality is present in the language.
  - An  $n$ -ary predicate  $P(x_1, \dots, x_n)$  (also called an  $n$ -place predicate).

Once again, the concrete meaning of each predicate must be specified in each structure. At the level of the language, it is just a symbol  $P$ , but we will allow ourselves to write it as  $P(x)$  or  $P(x_1, \dots, x_n)$  to make the number of places clear.

- **Functions.** At the end of the chapter, we will add functions to the language. For example,  $f(x)$  (which may represent  $f(x) = 2x + 3$  or  $f(x) = 1/x, \dots$ ), and  $f(x, y)$  (for  $f(x, y) = x + y$  or  $f(x, y) = x \times y, \dots$ ). Functions make the definition of a formula somewhat more technical, so for now our languages will not have functions.

### 3. Formulas

#### 3.1. Definitions

Because the language is rich, there is a specific vocabulary that must be mastered.

Let  $\mathcal{L}$  be a language without functions.

**Definition.**

A **term** of a language is a variable or a constant.

Remark: we will modify this definition for languages with functions (see the end of the chapter).

**Definition.**

An **atomic formula** is a word of the form  $P(t_1, \dots, t_n)$ , where  $P$  is an  $n$ -ary predicate and  $t_1, \dots, t_n$  are terms.

**Definition.**

The set of **formulas**  $\text{Form}(\mathcal{L})$  of the language is constructed according to the following rules:

- atomic formulas are formulas,
- if  $\mathcal{P}$  and  $\mathcal{Q}$  are formulas, then  $\neg \mathcal{P}$ ,  $(\mathcal{P} \wedge \mathcal{Q})$ ,  $(\mathcal{P} \vee \mathcal{Q})$ ,  $(\mathcal{P} \rightarrow \mathcal{Q})$ ,  $(\mathcal{P} \leftrightarrow \mathcal{Q})$  are also formulas,
- if  $x$  is a variable and  $\mathcal{P}$  is a formula, then
 
$$\forall x \mathcal{P} \quad \text{and} \quad \exists x \mathcal{P}$$
 are also formulas,
- no other words are formulas.

#### 3.2. Remarks

- The expression “ $\forall x \mathcal{P}$ ” is a word of the language, and in this language  $x$  is an abstract variable, not an element of a set. When we later interpret the formula, we will define a domain  $E$ , for example  $\mathbb{N}$ ,  $\mathbb{R}$ ,  $G$ , ... and then the formula will become “ $\forall x \in \mathbb{N} \mathcal{P}$ ” or “ $\forall x \in \mathbb{R} \mathcal{P}$ ,” etc.
- The interesting case is, of course, when the formula  $\mathcal{P}$  actually depends on the variable  $x$ . We will often allow ourselves to write a formula as  $\mathcal{P}(x)$  to show its dependence on the variable  $x$ . Thus, we will usually write “ $\forall x \mathcal{P}(x)$ ” or “ $\forall x \exists y \mathcal{P}(x, y)$ ”...
- As usual, we will allow ourselves to add and remove parentheses to make formulas clearer. For example, we will write “ $(\forall x \mathcal{P}) \rightarrow (\exists y \mathcal{Q})$ ” instead of the official formula “ $(\forall x \mathcal{P} \rightarrow \exists y \mathcal{Q})$ .”

The definition of a formula given above was the inductive definition (“from above”) of  $\text{Form}(\mathcal{L})$ . We can also give a recursive definition (“from below”).

We define  $\text{Form}(\mathcal{L}) = \bigcup_{n \geq 0} \text{Form}_n(\mathcal{L})$ , with:

$\text{Form}_0(\mathcal{L})$  is the set of atomic formulas

and then, for  $n \geq 0$ :

$$\begin{aligned} \text{Form}_{n+1}(\mathcal{L}) = & \text{Form}_n(\mathcal{L}) \\ & \cup \{ \neg \mathcal{P} \mid \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \{ (\mathcal{P} \wedge \mathcal{Q}) \mid \mathcal{P}, \mathcal{Q} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \dots \\ & \cup \{ \forall x_k \mathcal{P} \mid x_k \text{ a variable, } \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \\ & \cup \{ \exists x_k \mathcal{P} \mid x_k \text{ a variable, } \mathcal{P} \in \text{Form}_n(\mathcal{L}) \} \end{aligned}$$

where, of course, the union also includes the connectives  $\vee$ ,  $\rightarrow$ ,  $\leftrightarrow$ , and where  $x_1, x_2, \dots$  denotes the set of variables of  $\mathcal{L}$ .

## 4. Examples

Let us get a little ahead of the next chapter by looking at examples of structures in order to better understand the preceding notions.

### 4.1. With integers

We want to model the natural numbers with a zero and an order relation. We define a language  $\mathcal{L} = (c, P)$  with a constant  $c$  and a binary predicate  $P$ , also written  $P(x, y)$  (the equality predicate  $x = y$  is also included by default). Strictly speaking, we should also specify the names of the variables, but we will use the naming convention  $x, y, \dots$  throughout.

Here is a formula in the language:

$$\forall x \exists y \ P(x, y)$$

To use this language, we specify a domain, for example  $E = \mathbb{N}$  here. We must then define all the nonlogical symbols:

- the constant:  $c = 0$ ,
- the predicate:  $P(x, y)$ : “ $x < y$ .”

The logical symbols, on the other hand, do not need to be specified: the variable  $x$  remains a variable, the connective  $\wedge$  remains AND, etc. We have thus defined a **structure**, denoted by  $\mathcal{S}$ . In this structure, the preceding formula is interpreted as:

$$\forall x \in \mathbb{N} \exists y \in \mathbb{N} \ x < y,$$

that is, “for every natural number, there exists an even larger natural number.”

For the same language  $\mathcal{L}$ , we can make other choices of structure, and the same formula is interpreted differently.

Here are a few examples illustrating the vocabulary associated with formulas, each time giving a word of the language and its interpretation in the structure.

Let us begin with terms: there are only the constant  $c$  and the variables  $x, y, \dots$ . In the structure, we replace  $c$  with 0.

| Term in $\mathcal{L}$ | Interpretation in $\mathcal{S}$ |
|-----------------------|---------------------------------|
| $c$                   | 0                               |
| $x$                   | $x$                             |

Atomic formulas are formed using a predicate. Here, only the predicate  $P$  is defined, and it is interpreted as  $x < y$  in the structure. We also always have the equality predicate “ $x = y$ .”

| Atomic formula in $\mathcal{L}$ | Interpretation in $\mathcal{S}$ |
|---------------------------------|---------------------------------|
| $P(c, x)$                       | $0 < x$                         |
| $P(c, c)$                       | $0 < 0$                         |
| $P(x, y)$                       | $x < y$                         |
| $P(y, x)$                       | $y < x$                         |
| $x = c$                         | $x = 0$                         |

For formulas, connectives and quantifiers are allowed. A formula “ $\forall x \mathcal{P}$ ” is interpreted in the structure as “ $\forall x \in \mathbb{N} \mathcal{P}$ .”

| Formula in $\mathcal{L}$                                                           | Interpretation in $\mathcal{S}$                                                                                                 |
|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| $\forall x \ P(c, x)$                                                              | $\forall x \in \mathbb{N} \ 0 < x$                                                                                              |
| $\exists y \ x = y$                                                                | $\exists y \in \mathbb{N} \ x = y$                                                                                              |
| $\forall x \ P(c, x) \vee (x = c)$                                                 | $\forall x \in \mathbb{N} \ (0 < x) \vee (x = 0)$                                                                               |
| $\forall x \ \forall y \ P(x, y) \rightarrow (\exists z \ P(x, z) \wedge P(z, y))$ | $\forall x \in \mathbb{N} \ \forall y \in \mathbb{N} \ (x < y) \rightarrow (\exists z \in \mathbb{N} \ (x < z) \wedge (z < y))$ |

For the integers, the last formula is interpreted as “between any two distinct integers, one can find an integer strictly between them.” This is clearly false (for example, there is no integer strictly between 7 and 8). It would be true for another structure whose domain is the set of real numbers  $\mathbb{R}$  (between two distinct real numbers, there are even infinitely many real numbers).

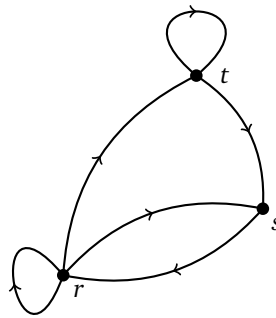
## 4.2. With graphs

We want to model graphs (such as the one in the figure below). We define a language  $\mathcal{L}$  with the binary predicate  $R$  (and no constants).

In this language, the only terms are variables, for example  $x$  and  $y$ . The atomic formulas are  $R(x, x)$ ,  $R(x, y)$ ,  $R(y, x)$ ,  $R(y, y)$ , as well as  $x = x$ ,  $x = y$ ,  $y = x$ ,  $y = y$ . Here is an example of a formula:

$$\forall x \ \exists y \ R(x, y)$$

We have our language. How do we model a graph? Here is the directed graph we want to model:



There are three vertices, so we define a structure whose domain is the three-element set  $E = \{r, s, t\}$ . In the structure, it is the predicate that describes the edges: we interpret the predicate  $R$  by saying that  $R(x, y)$  is true if there is an edge from  $x$  to  $y$  (we say that  $x$  is related to  $y$ ). Thus, in our example, we define the predicate in the structure by:

$$R(r, s) = \text{T}, \quad R(s, t) = \text{F}, \quad R(t, s) = \text{T}, \dots$$

The preceding formula in the language  $\mathcal{L}$  is interpreted in this structure as:

$$\forall x \in \{r, s, t\} \quad \exists y \in \{r, s, t\} \quad R(x, y)$$

In other words, “for every vertex  $x$ , there is a vertex  $y$  for which there is an edge from  $x$  to  $y$ ”; put differently, “at least one edge leaves every vertex,” which is true for the graph under consideration.

Here are some sentences in the language. It is up to you to determine what they mean for graphs and whether they are true or false for the example above.

- $\forall x \forall y \quad R(x, y)$
- $\exists x \exists y \quad R(x, y)$
- $\exists x \forall y \quad R(x, y)$
- $\forall x \forall y \quad R(x, y) \vee R(y, x)$
- $\exists x \exists y \quad R(x, y) \wedge R(y, x)$

### 4.3. There exists exactly one; there exist two

- “ $\exists x P(x)$ ” requires the existence of at least one  $x$  for which  $P(x)$  holds. There may be one or more.
- Here is how to express that there is exactly one element for which  $P(x)$  holds:

$$(\exists x \quad P(x)) \wedge (\forall x \forall y \quad (P(x) \wedge P(y) \rightarrow x = y))$$

On the one hand, we require an element to exist; on the other, if there is another one, it is the same element. We will not use the notation “ $\exists!$ ”.

- Here is how to express that there are at least two elements:

$$\exists x \exists y \quad (x \neq y) \wedge P(x) \wedge P(y)$$

We write  $x \neq y$  for  $\neg(x = y)$ .

## 5. Language with functions

### 5.1. Functions

A **function**  $f$  takes one or more objects as input and returns an object as output. Functions must not be confused with predicates: predicates also take objects as input, but they return a T/F value.

Here are some examples of interpretations of functions:

- $f(x) = x^2$ , a function of one variable (here, over the real numbers),
- $f(x, y) = x \% y$ , the remainder in the Euclidean division of  $x$  by  $y$  (here, over the integers).

In contrast,  $P(x)$ : “ $x = 0$ ” and  $Q(x, y)$ : “ $x < y$ ” are predicates, not functions.

From now on, we allow  $n$ -ary functions (or functions of  $n$  arguments) to be added as nonlogical symbols of the language. As with predicates, we will often abuse notation and write functions as  $f(x)$ ,  $f(x, y)$ ,  $f(x_1, \dots, x_n)$ , even though the symbol in the language is just  $f$ .

## 5.2. Formulas

With the introduction of functions, we need to adapt our definition of a formula.

Let  $\mathcal{L}$  be a language (with or without functions).

**Definition.**

The **terms** of a language are:

- the variables and constants,
- and if  $t_1, \dots, t_n$  are terms and  $f$  is an  $n$ -ary function symbol, then  $f(t_1, \dots, t_n)$  is also a term.

Note that the definition is recursive.

The definition of an **atomic formula** is unchanged: it is an expression of the form  $P(t_1, \dots, t_n)$ , where  $P$  is an  $n$ -ary predicate and  $t_1, \dots, t_n$  are terms (according to our new definition).

The definition of a **formula** is unchanged: it is the same construction using atomic formulas, connectives, and quantifiers (except, of course, that terms and atomic formulas may now contain functions).

## 5.3. Examples

Let  $\mathcal{L}$  be the language consisting of:

- constants  $a, b$ ,
- a binary predicate  $P$ , written (by an abuse of notation) as  $x < y$ ,
- a unary function  $f$  and a binary function  $g$ .

- Examples of terms:

$$a, \quad y, \quad f(x), \quad g(a, y), \quad g(f(x), b), \quad f(g(y, x)), \quad \dots$$

- Examples of atomic formulas:

$$x < a, \quad f(x) < f(y), \quad g(a, f(b)) < x, \quad \dots$$

- Examples of formulas:

$$\neg(x < y), \quad (f(x) < a) \wedge (f(y) < b), \quad \forall x \, f(x) < b, \quad \exists x \, g(x, y) = a, \\ \forall x \, (f(x) < a) \rightarrow (\exists y \, g(x, y) < b)$$

As before, we anticipate the notion of a structure in order to make these functions concrete. For example, we may work over the set of real numbers  $\mathbb{R}$  and consider the following interpretation of the nonlogical symbols:

- $a = 0, b = 1$ ,
- $P(x, y)$  denotes the usual strict inequality  $x < y$ ,
- $f(x) = x^2$  and  $g(x, y) = x + y$ .

- Examples of terms in this interpretation:

$$x + 1, \quad y^2, \quad x^2 + y, \quad (x + y)^2, \quad x + (y^2 + 1)^2, \quad \dots$$

- Examples of atomic formulas in this interpretation:

$$x < 0, \quad x + y < x^2, \quad x^2 + y^2 = 1, \quad \dots$$

- Examples of formulas in this interpretation:

$$(x^2 + y^2 < 1) \rightarrow ((x < 1) \wedge (y < 1)), \quad \forall x \, x^2 < x + y, \quad (y < 1) \rightarrow (\exists x \, x^2 = y), \quad \dots$$

## Exercises

### Language, Formulas, Examples

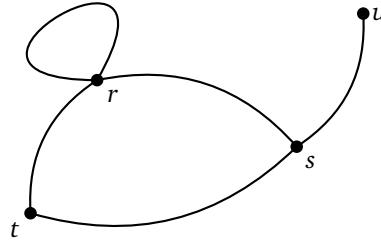
**Exercise 1.** Consider the language  $\mathcal{L}$  consisting of the constants  $a$  and  $b$ , a unary predicate  $P$ , and a binary predicate  $Q$  (as well as the variables  $x, y$ , the connectives, and the quantifiers). Determine whether the following words are terms, atomic formulas, or formulas.

$$\neg a, \quad xy, \quad (x \wedge a), \quad P(a), \quad \neg Q(x) \\ \forall x Q(x, y), \quad \exists y P(x), \quad \forall a \exists Q(a, b) \\ \forall x \exists y P(x \rightarrow y), \quad \exists x ((\forall y Q(x, y)) \vee (\forall y \neg Q(x, y)))$$

**Exercise 2.** Consider the language  $\mathcal{L}$  equipped with a constant  $c$  and a binary predicate  $P(x, y)$ . Determine whether the following words are terms, atomic formulas, or formulas. If they are, give their interpretation in the structure with domain  $\mathbb{N}$ , where  $c = 0$  and  $P(x, y)$  is defined by  $x < y$ . Then interpret them in the structure with domain  $\mathbb{R}$ , where  $c = 1$  and  $P(x, y)$  is defined by  $x \leq y$ .

$$P, \quad c^2, \quad P(c, c), \quad \neg P(c, 0), \quad P(x, P(y, c)), \quad P(x = c) \rightarrow x = c \\ P(x, c), \quad \exists x, \quad x > c \rightarrow \forall y P(y, c), \quad \forall y (P(x, c) \rightarrow y = x) \\ \exists x \vee y P(x, c) \vee P(y, c), \quad \forall x ((x = c) \leftrightarrow (\exists y P(y, c)))$$

**Exercise 3.** Consider a language with a binary predicate  $R(x, y)$ . Consider the structure with domain  $E = \{r, s, t, u\}$ , and interpret  $R(x, y)$  as true if there is an (undirected) edge between the vertices  $x$  and  $y$  of a graph  $G$  (such as the one below).



Translate the following sentences into the language  $\mathcal{L}$ :

1. There is a vertex connected to itself.
2. There is a vertex connected to every other vertex.
3. Every vertex is connected to at least one other vertex.
4. There is a vertex that is not connected to any other vertex.
5. Not all the vertices are connected to one another.
6. There is exactly one vertex connected to itself.

**Exercise 4.** Consider a language  $\mathcal{L}$  equipped with a predicate  $P(x)$  and equality  $x = y$ . Translate the following sentences into formulas of  $\mathcal{L}$ :

1. No element satisfies  $P$ .
2. There is at most one element that satisfies  $P$ .

3. There are exactly two elements that satisfy  $P$ .
4. There are at least three elements that satisfy  $P$ .
5. All but one element satisfy  $P$ .

## Language with functions

**Exercise 5.** Consider a language  $\mathcal{L}$  having constants  $a, b$ , a predicate  $P(x)$ , and two functions  $f(x)$  and  $g(x, y)$ . Determine whether the following words are terms, atomic formulas, or formulas of  $\mathcal{L}$ . If they are, give their interpretation in the structure with domain  $\mathbb{R}$ , where  $a = 0$ ,  $b = 1$ ,  $P(x)$  is defined by  $x \geq 0$ ,  $f(x) = \cos(x)$ , and  $g(x, y) = x \times y$ .

$$\begin{aligned}
 &f(x), \quad g(a, b), \quad f(P(a)), \quad \neg P(f(a)), \quad P(g(y, x)), \quad P(g(f(x), f(f(b)))) \\
 &P(x) \wedge P(y) \rightarrow g(x, y), \quad \exists x \, f(x) = b, \quad \forall y \, P(y) \leftrightarrow P(f(y)) \\
 &\forall x \forall b \, P(g(x, b)) \rightarrow (P(x) \wedge P(b)), \quad \exists x \, (P(x) \leftrightarrow (\forall y \, (P(y) \rightarrow P(g(x, y))))))
 \end{aligned}$$

# Variables and Structures

*Our goal is to use a logical language to describe concrete mathematical situations. To do so, we introduce the notion of a structure, which connects abstract formulas with actual mathematical objects. Before deciding whether, in a given structure, a formula is true or false, it is essential to distinguish free variables from bound variables.*

## 1. Free variables, bound variables

### 1.1. Idea

Consider a language  $\mathcal{L}$  with variables  $x, y, z, x_1, x_2, \dots$ . In a given formula, a variable may be free or bound. Let us understand the difference through examples before giving the precise definition.

Consider a formula “ $\forall x \ P(x)$ ” or “ $\exists x \ P(x)$ ”. Here the variable “ $x$ ” is *quantified*, that is, *bound* by a quantifier. Likewise, the two variables  $x$  and  $y$  in “ $\forall x \ \exists y \ Q(x, y)$ ” are quantified.

By contrast, in the formula “ $\forall x \ Q(x, y)$ ”, the variable  $y$  is not quantified; we say that this variable is *free* (but the formula is still legitimate).

Here is a formula (under an interpretation):

$$\forall x \in \mathbb{R} \quad \exists y \in \mathbb{R} \quad y > x$$

Since there are no free variables, we can assign this formula a truth value T/F.

By contrast:

$$\forall x \in \mathbb{R} \quad y > x$$

is a formula in which  $y$  is a free variable. It makes no sense to ask whether this formula is true or false until a value has been assigned to  $y$ .

### 1.2. Definition

#### Scope

#### Definition.

In a formula containing “ $\forall x \ P$ ” or “ $\exists x \ P$ ”, the *scope* of  $\forall$  (or  $\exists$ ) is  $P$ .

It is worth noting that  $P$  may be an arbitrarily complex formula.

Example:

$$\underbrace{\forall x \quad \exists y \quad \underbrace{P(x) \rightarrow Q(x, y)}_{\text{scope of } \exists}}_{\text{scope of } \forall}$$

#### Free or bound variable

**Definition.**

In a formula, a variable  $x$  is a **bound variable** if every occurrence of  $x$  lies within the scope of a quantifier  $\forall x$  or  $\exists x$ . Otherwise,  $x$  is called a **free variable**.

Examples:

- $\forall x \quad P(x) \vee Q(x, y)$ :  $x$  is bound,  $y$  is free.
- $P(x) \rightarrow Q(x, y)$ :  $x$  and  $y$  are free.
- $(\forall x \quad P(x)) \leftrightarrow (\exists y \quad P(y))$ :  $x$  and  $y$  are bound.
- $(\forall x \quad x = y) \vee (\exists y \quad x = y)$ :  $x$  and  $y$  are free!

There is an important subtlety here, illustrated by this last example. For a variable  $x$  to be free, it is enough for at least one occurrence of  $x$  not to lie within the scope of a quantifier. In our example,  $y$  is a free variable in the left-hand part and  $x$  is free in the right-hand part, so  $x$  and  $y$  are indeed free variables in the formula as a whole.

**Closed formula****Definition.**

A formula is **closed** if it contains no free variables.

It is also called a **closed formula**, or a **sentence**.

The idea is that we will be able to assign a truth value T/F to a closed formula, whereas a formula that is not closed still has free variables to which values must be assigned.

Examples:

- $\forall x \quad (P(x) \wedge (\exists y \quad P(y)))$  is a closed formula.
- $\exists x \quad ((\exists y \quad P(x) \wedge P(y)) \rightarrow (\forall y \quad Q(x, y)))$  is also a closed formula.
- $\exists x \quad (Q(x, y) \rightarrow (\exists z \quad P(z)))$  is not closed because  $y$  is a free variable.
- $(\forall x \quad P(x)) \vee (\exists x \quad Q(x)) \vee (\neg R(x))$  is not a closed formula because  $x$  is a free variable (in the right-hand subformula).

**1.3. Substitution**

In a language  $\mathcal{L}$ , let  $\mathcal{P}$  be a formula,  $x$  a variable, and  $t$  a term.

**Definition.**

$\mathcal{P}[t/x]$  denotes the formula in which every free occurrence of the variable  $x$  has been replaced by the term  $t$ .

Remark: recall that terms include constants and variables, and, if the language contains a function symbol  $f$ , then  $f(t_1, \dots, t_n)$  is also a term whenever  $t_1, \dots, t_n$  are terms.

**Example.**

Let  $\mathcal{L}$  be a language containing constants  $a, b$ , a binary function denoted by  $+$ , a unary predicate  $P$ , and a binary predicate  $Q$ .

Let us study the formula:

$$\mathcal{P} : \quad \forall y \quad P(x) \rightarrow Q(x, y)$$

- $t = a, \mathcal{P}[t/x] : \forall y \quad P(a) \rightarrow Q(a, y)$

- $t = x + b, \mathcal{P}[t/x] : \forall y \quad P(x + b) \rightarrow Q(x + b, y)$
- $t = b, \mathcal{P}[t/y] : \forall y \quad P(x) \rightarrow Q(x, y)$ : the formula does not change because every occurrence of  $y$  lies within the scope of a quantifier.

Remark: the formal definition of substitution is given by induction, beginning with atomic formulas and then introducing the logical connectives. For formulas with quantifiers “ $\forall x \mathcal{P}$ ” or “ $\exists x \mathcal{P}$ ”, no substitution for the variable  $x$  is made because the variable is quantified.

We can also perform a **simultaneous substitution**  $\mathcal{P}[t_1/x_1, t_2/x_2, \dots, t_n/x_n]$ . It is denoted by  $\mathcal{P}[s]$ , where  $s$  denotes  $(t_1/x_1, t_2/x_2, \dots, t_n/x_n)$ . It is important to understand that the substitutions are simultaneous: every free occurrence of  $x_1$  is replaced by  $t_1$  and, *at the same time*, every free occurrence of  $x_2$  is replaced by  $t_2$ , and so on.

### Example.

Let  $\mathcal{P}$  be the formula “ $P(x, y)$ ”. Then:

- $\mathcal{P}[a/x, b/y] : P(a, b)$
- $\mathcal{P}[x + y/x, x/y] : P(x + y, x)$
- $\mathcal{P}[y/x, x/y] : P(y, x)$

Remark: the formula  $\mathcal{P}$ : “ $P(x) \rightarrow (\forall x \quad Q(x))$ ” is a legitimate formula of the language (it is as though we had two different  $x$ ’s: one red and one blue). Substituting  $a$  for  $x$  gives  $\mathcal{P}[a/x]$ : “ $P(a) \rightarrow (\forall x \quad Q(x))$ ”, because only the free occurrences of  $x$  are replaced. We would prefer to write the initial formula as  $\mathcal{P}'$ : “ $P(x) \rightarrow (\forall y \quad Q(y))$ ”, which is easier to understand.

### Variable capture.

When performing a substitution  $\mathcal{P}[t/x]$ , we always assume that no variable occurring in  $t$  falls within the scope of a quantifier in  $\mathcal{P}$ . If the term  $t$  contains variables that might become bound by quantifiers in  $\mathcal{P}$ , the substitution could change the logical meaning of the formula: this is called variable capture. For example, let:

$$\mathcal{P} : \quad \forall y \quad P(x) \rightarrow Q(x, y)$$

Substituting  $t = y$  for  $x$  is not legitimate, because it would give the formula:

$$\mathcal{P}[y/x] : \quad \forall y \quad P(y) \rightarrow Q(y, y)$$

which does not have the same meaning at all.

To avoid this phenomenon, we first rename the bound variables of  $\mathcal{P}$  before performing the substitution. In our example, we first rewrite  $\mathcal{P}$  in the following form:

$$\mathcal{P} : \quad \forall z \quad P(x) \rightarrow Q(x, z)$$

and then substitute  $y$  for  $x$ :

$$\mathcal{P}[y/x] : \quad \forall z \quad P(y) \rightarrow Q(y, z)$$

## 2. Structures

A language remains an abstract shell; it provides a framework for a wide variety of mathematical objects and their properties: integers, groups, vector spaces, set theory, and so on.

### 2.1. Definition

#### Definition.

Consider a language  $\mathcal{L}$ . A **structure**  $\mathcal{S}$  for this language consists of:

- a nonempty set  $E$ , called the **domain**,
- an element  $c^{\mathcal{S}} \in E$  associated with each constant symbol  $c$  of  $\mathcal{L}$ ,
- a subset  $P^{\mathcal{S}} \subset E^n$  associated with each  $n$ -place predicate symbol  $P$ , in other words, a function  $\phi_P : E^n \rightarrow \{T, F\}$  such that  $\phi_P(a_1, \dots, a_n)$  is true if and only if  $(a_1, \dots, a_n) \in P^{\mathcal{S}}$ ,
- a function  $f^{\mathcal{S}} : E^n \rightarrow E$  associated with each  $n$ -place function symbol  $f$ .

$\mathcal{S}$  is also called an **interpretation** of the language  $\mathcal{L}$ . Several different structures can be associated with the same language.

A structure is therefore a concrete mathematical realization of a language. To define a structure, we must define all the nonlogical symbols: constants, predicates, and functions. The logical symbols are left unchanged: a variable  $x$  remains a variable  $x$ , except that it now ranges over the domain  $E$ .

### 2.2. A first example

Consider a language  $\mathcal{L} = (c, P, f)$ , where  $c$  is a constant,  $P(x, y)$  a binary predicate, and  $f(x)$  a unary function. As usual, we do not explicitly state that the variables are  $x, y$ , that the logical connectives are present, and that there is also the equality predicate “ $x = y$ ”.

We will define a structure that corresponds to the natural numbers and models the successor. Consider the structure  $\mathcal{S}$  with domain  $E = \mathbb{N}$  and interpretation  $(0, <, x \mapsto x + 1)$  of the language, that is:

- for the constant  $c$  of  $\mathcal{L}$ , define  $c^{\mathcal{S}} = 0$ ; this is the interpretation of the symbol  $c$  in  $\mathcal{S}$ ,
- for the predicate  $P$  of  $\mathcal{L}$ , define  $P^{\mathcal{S}}$  by  $P^{\mathcal{S}}(x, y)$ :  $x < y$ ,
- for the function  $f$  of  $\mathcal{L}$ , define  $f^{\mathcal{S}}$  by  $f^{\mathcal{S}}(x)$ : the successor of the natural number  $x$ , denoted by  $x + 1$ .

Once the structure has been defined, every formula of  $\mathcal{L}$  can be interpreted as a mathematical statement about the structure. Here are some examples:

| Formula of $\mathcal{L}$      | Formula interpreted in $\mathcal{S}$                      | True or false in $\mathcal{S}$ ?                              |
|-------------------------------|-----------------------------------------------------------|---------------------------------------------------------------|
| $\forall x \exists y P(x, y)$ | $\forall x \in \mathbb{N} \exists y \in \mathbb{N} x < y$ | It is true                                                    |
| $\forall x P(c, f(x))$        | $\forall x \in \mathbb{N} 0 < x + 1$                      | It is also true                                               |
| $\forall x f(x) = y$          | $\forall x \in \mathbb{N} x + 1 = y$                      | The question makes no sense<br>because $y$ is a free variable |

When defining this structure  $\mathcal{S}$ , we will often write, somewhat abusively,  $c = 0$ ,  $P(x, y)$ :  $x < y$ , and  $f(x) = x + 1$  (instead of  $c^{\mathcal{S}}$ ,  $P^{\mathcal{S}}$ , and  $f^{\mathcal{S}}$ ).

There are of course many other possible structures for this same language. We can change the domain, the constant, the predicate, or the function:

- Structure  $\mathcal{S}_1$  with domain  $\mathbb{R}$ , again with  $(0, <, x \mapsto x + 1)$ , except that this time the interpreted formula  $\forall x \in \mathbb{R} \quad 0 < x + 1$  is false.
- Structure  $\mathcal{S}_2$  with domain  $\mathbb{R}$ , with  $(0, \leq, x \mapsto x^2)$ : the formula  $\forall x \quad P(c, f(x))$  of the language becomes  $\forall x \in \mathbb{R} \quad 0 \leq x^2$ , a formula that is true under this interpretation.

## 2.3. Comments

Here are some important remarks.

### Nonempty domain

The domain of a structure must be a nonempty set. This may seem like a detail, but it is quite important. It seems natural to everyone that:

If “ $\forall x \in E \quad P(x)$ ” is true, then “ $\exists x \in E \quad P(x)$ ” is also true.

But suppose for a moment that  $E$  were the empty set. Then “ $\forall x \in E \quad P(x)$ ” would always be true (since there is nothing to check, because there is no  $x$  to consider). By contrast, “ $\exists x \in E \quad P(x)$ ” would always be false (because  $E$  is empty). This situation would be problematic.

We can also argue that if  $E$  were empty, then “ $\forall x \in E \quad x \neq x$ ” would be true and “ $\exists x \in E \quad x = x$ ” would be false, which would be very troublesome.

Conclusion: we always assume that the domain  $E$  is nonempty.

### Functions

Let us return briefly to functions. In the language,  $f$  is merely a function symbol that must be followed by  $n$  terms (for an  $n$ -place function). In the structure, we must define the interpretation of this function using a good old-fashioned function:  $f^{\mathcal{S}} : E^n \rightarrow E$ . It is a function of  $n$  variables in the usual sense. As already mentioned, in practice we simply denote it by  $f : E^n \rightarrow E$ .

For example, let  $\mathcal{L}$  be a language with a unary function  $f$  and a binary function  $g$ .

- For a domain  $\mathbb{N}$ , we could define  $f$  by  $f(x) = x + 1$  or  $f(x) = x^2$ , and  $g$  by  $g(x, y) = x + y$  or  $g(x, y) = xy$ . By contrast, we cannot define  $f$  by  $f(x) = x - 1$ , because  $f(0)$  is not in the domain  $\mathbb{N}$ , nor can we define  $g$  by  $g(x, y) = x/y$ , which does not always yield integers.
- On the domain  $] - \infty, 0[$ , however, we could define  $f(x) = x - 1$  and  $g(x, y) = -x/y$ .

### Predicates

The interpretation of a predicate may be a little more difficult to understand. In the language, an  $n$ -ary predicate is merely a symbol that must be followed by  $n$  terms. For the structure, we must define each predicate, that is, specify for which objects its value is True.

Consider the example of a unary predicate  $P$  in a language  $\mathcal{L}$ . To define a structure  $\mathcal{S}$ , we define its domain  $E$  and then the interpretation of the predicate, that is, all the values in  $E$  that make  $P(x)$  true. We can do this by defining a subset  $P^{\mathcal{S}} \subset E$ , which is exactly the set of all  $d \in E$  such that  $P(d)$  is true. We can also define it as a function  $\phi_P : E \rightarrow \{T, F\}$  that returns True or False. The connection between the two is that  $\phi_P(d) = T$  if and only if  $d \in P^{\mathcal{S}}$ .

Thus, if we want to define an interpretation of “being a real number greater than or equal to 7”, we set  $E = \mathbb{R}$  and, for the predicate, either define  $P^{\mathcal{S}} = [7, +\infty[$ , the set of values that satisfy the property, or define a function:

$$\begin{aligned} \phi_P : E &\longrightarrow \{T, F\} \\ x &\longmapsto \begin{cases} T & \text{if } x \in [7, +\infty[ \\ F & \text{otherwise.} \end{cases} \end{aligned}$$

Remark: we emphasized that a domain must be nonempty. By contrast, we do not require a predicate to be realized. In other words,  $P$  may always evaluate to false in the structure. Thus, over  $\mathbb{R}$ , we are allowed to define the predicates “ $x^2 < 0$ ” or “ $x \neq x$ ”, even though no  $x \in \mathbb{R}$  will realize them. To define a binary predicate  $P(x, y)$  in a structure with domain  $E$ , we must define a set  $P^S \subset E^2$ , which will be the set of pairs  $(d_1, d_2)$  such that  $P(d_1, d_2)$  is true.

## Domain

When we define a structure, we fix a domain  $E$ . The variables are then interpreted as objects in this domain. For example, the formula of the language “ $\forall x \exists y \quad Q(x, y)$ ” can be interpreted in a structure where  $E = \mathbb{N}$  as “ $\forall x \in \mathbb{N} \exists y \in \mathbb{N} \quad x < y$ ”. But then how can we write a formula in which  $x$  and  $y$  do not range over exactly the same set, such as “ $\forall \varepsilon > 0 \exists n \in \mathbb{N} \quad n > \frac{1}{\varepsilon}$ ”? We need to use a small trick. Let  $A$  and  $B$  be two subsets of the same set  $E$ . We would like to write the mathematical statement:

$$\forall x \in A \quad \exists y \in B \quad Q(x, y)$$

It cannot come directly from a formula of the language, because  $x$  and  $y$  must range over the same set. Define a predicate  $P_A$  such that  $P_A(x)$  is true exactly when  $x \in A$ . Likewise, define  $P_B$  so that  $P_B(x)$  is true when  $x \in B$ . Consider the formula of the language:

$$\forall x \quad (P_A(x) \rightarrow \exists y (P_B(y) \wedge Q(x, y)))$$

It is interpreted in our structure with domain  $E$  (with  $P_A$  and  $P_B$  defined as above) as:

$$\forall x \in E \quad (x \in A \rightarrow \exists y \in E (y \in B) \wedge Q(x, y))$$

If  $x \notin A$ , then this statement is true (because the implication is true by definition). If  $x \in A$ , then for this statement to be true, there must exist a  $y \in B$  such that  $Q(x, y)$  is true. This is indeed what we wanted mathematically.

## 2.4. Other examples

### Inverse

We want to express the existence of an inverse (or an additive inverse). This is a very general notion that appears in many situations and is therefore worth conceptualizing. Let  $\mathcal{L}$  be the language with a constant  $e$  and a binary function  $f(x, y)$ , which we denote by  $x \circ y$ . We can write a formula expressing the existence of a (right) inverse:

$$\forall x \quad \exists y \quad x \circ y = e$$

Such a  $y$  is called a (*right*) *inverse* of  $x$ .

Concretely, here is what this covers:

- **Additive inverse.** Define the structure  $\mathcal{S}$  with domain  $\mathbb{Z}$ , with  $e^S = 0$  and  $f^S(x, y) = x + y$ . The formula of the language becomes:

$$\forall x \in \mathbb{Z} \quad \exists y \in \mathbb{Z} \quad x + y = 0$$

In other words,  $y = -x$  is the *additive inverse* of  $x$  here.

- **Multiplicative inverse.** This time define the structure  $\mathcal{S}$  with domain  $\mathbb{R}^*$ , with  $e^S = 1$  and  $f^S(x, y) = x \times y$ . The formula of the language becomes:

$$\forall x \in \mathbb{R}^* \quad \exists y \in \mathbb{R}^* \quad x \times y = 1$$

In other words,  $y = \frac{1}{x}$  is the *inverse* of  $x$  here (in the algebraic sense).

- **Matrix inverse.** Consider the set of invertible  $n \times n$  matrices, equipped with matrix multiplication, and let  $e^S = I$  be the identity matrix. In this structure, the formula expresses  $A \times B = I$ , that is,  $B = A^{-1}$  is the inverse of the matrix  $A$ .

- **And so on.** We could continue with a group  $(G, \diamond)$  with identity element  $e$ , to express  $g \diamond h = e$ ; or with the notion of the inverse bijection of a function, to express  $g \circ h = \text{id}$ ; and so on. All of this uses the same language  $\mathcal{L}$ .

There are of course other formulas of the language that are of interest for all these structures. Associativity:

$$\forall x \quad \forall y \quad \forall z \quad x \circ (y \circ z) = (x \circ y) \circ z$$

The action of the identity element, which changes nothing:

$$\forall x \quad ((e \circ x) = x) \wedge ((x \circ e) = x)$$

## Graph

Let us illustrate how the same formula can become true or false depending on the chosen structure. Consider the language  $\mathcal{L}$  with the predicate  $R(x, y)$ , intended to model undirected graphs.

Consider the formula of  $\mathcal{L}$ :

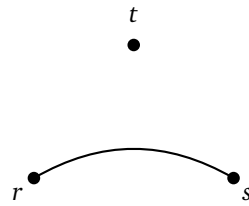
$$\forall x \quad \exists y \quad (x \neq y) \wedge R(x, y)$$

which should be understood as “every vertex is connected to at least one other vertex”.

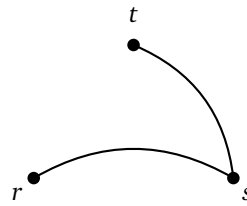
- Define  $S_1$  with domain  $E = \{r, s, t\}$ , where  $R^{S_1}$  is defined by the graph  $G_1$  on the left below; that is,  $R$  is true only for the pairs  $(r, s)$  and  $(s, r)$  (which are connected to one another).

In this structure, the interpreted formula is false because  $t$  is not connected to any vertex.

- If, on the other hand, we define  $S_2$  with domain  $E = \{r, s, t\}$ , where the predicate is now defined by the graph  $G_2$  on the right, then the interpreted formula is true.



Graph  $G_1$



Graph  $G_2$

## 2.5. Advanced examples

### Rings

Consider a language  $\mathcal{L} = (0, 1, +, -, \times)$ , where  $0, 1$  are constants and  $+, -, \times$  are binary functions (strictly speaking, we should write these symbols as  $a, b, f, g, h$ , because here  $0$  is only a symbol of the language, named in anticipation of its interpretation). This language is suited to modeling the notion of a ring (a set with addition, subtraction, and multiplication, but not necessarily inverses).

Here are some examples of structures:

- **Integers.** Consider the structure  $\mathcal{S}$  with domain  $\mathbb{Z}$ . The constants are naturally interpreted as the integers  $0$  and  $1$ , and the functions are interpreted as the usual operations on integers.
- **Polynomials.** For this structure  $\mathcal{S}$ , the domain is the set of polynomials  $\mathbb{R}[X]$ ; this time  $0$  denotes the zero polynomial,  $1$  the constant polynomial equal to  $1$ , and the operations are polynomial addition, subtraction, and multiplication.
- **Matrices.** For this new structure, the domain is the set  $M_n(\mathbb{R})$  of  $n \times n$  matrices. Naturally,  $0$  denotes the zero matrix,  $1$  denotes the identity matrix  $I$ , and the operations are matrix operations.

The point of the language is therefore to formalize properties that apply in many situations. For a ring, we additionally require the following formulas to be true once interpreted in the structure:

- $\forall x \quad x - x = 0$ : subtraction corresponds to taking the additive inverse,
- $\forall x \quad 1 \times x = x$ : 1 is an identity element for multiplication,
- $\forall x \quad \forall y \quad \forall z \quad x \times (y + z) = x \times y + x \times z$ : distributivity,
- and so on.

### Set theory

Consider a language  $\mathcal{L} = (\in)$ , where the membership symbol “ $\in$ ” denotes a binary predicate  $P(x, y)$ , written here as  $x \in y$ .

Warning: in the following structures, the objects will be sets, and the predicate  $x \in y$  will ask whether the set  $x$  is indeed an *element* of the set  $y$ . For example,  $\{1\} \in \{\emptyset, \{1\}, \{2\}, \{1, 2\}\}$ : a set is indeed an element of another set. The symbol  $\in$  will not denote the usual membership  $1 \in \{1, 2, 3\}$  (since 1 and  $\{1, 2, 3\}$  are different kinds of objects, one being an integer and the other a set of integers), nor the inclusion  $\{1, 2\} \subset \{1, 2, 3\}$ , which denotes something else.

Let us construct a first structure  $S_1$ . Its domain is  $E = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\{\{\emptyset\}\}\}, \dots\}$ ; every element is a singleton except the empty set. The predicate is defined by the following relations:

$$\emptyset \in \{\emptyset\} \quad \{\emptyset\} \in \{\{\emptyset\}\} \quad \dots$$

No other membership relations hold.

Formally, define  $E = \bigcup_{n \geq 0} E_n$ , where:

$$E_0 = \{\emptyset\} \quad \text{and} \quad E_{n+1} = E_n \cup \{\{x\} \mid x \in E_n\}$$

Thus,  $E_1$  is formed from  $\emptyset$  and  $\{\emptyset\}$ , so  $E_1 = \{\emptyset, \{\emptyset\}\}$ . Then  $E_2$  is formed by the union of  $E_1$  and  $\{\{\emptyset\}, \{\{\emptyset\}\}\}$ , so  $E_2 = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}\}$  (we do not count  $\{\emptyset\}$  twice).

To define the predicate in this structure, we give the set of  $(x, y) \in E^2$  such that  $x \in y$ :

$$\in^S : \quad \{(x, \{x\}) \mid x \in E\}$$

In other words,  $x \in \{x\}$  and no other membership relation holds. In this structure, if  $x \in y$ , then  $y = \{x\}$ . The set  $\{x\}$  is called the *successor* of  $x$ .

What could such a complicated construction possibly be used for? If we write  $\emptyset = 0$ ,  $\{\emptyset\} = 1$ , and  $\{\{\emptyset\}\} = 2$ , then 1 is the successor of 0, 2 is the successor of 1, and so on. We have constructed a model for the set of natural numbers  $\mathbb{N}$ , whose only property here is that every natural number has a successor.

When constructing a set theory, we add axioms or additional notions. Here are some formulas that we require to be true when interpreted in the structure:

- Equality of two sets. Two sets are equal if and only if they have the same elements:

$$\forall x \quad \forall y \quad (x = y) \leftrightarrow (\forall z \quad (z \in x \leftrightarrow z \in y))$$

- Empty set axiom:

$$\exists x \quad \forall y \quad \neg(y \in x)$$

That is, there exists a set containing no elements.

- Inclusion is defined by:

$$x \subset y \quad \text{means by definition} \quad \forall z \quad (z \in x \rightarrow z \in y)$$

- Union axiom:

$$\forall x \quad \exists u \quad \forall z \quad (z \in u) \leftrightarrow (\exists y \quad (y \in x) \wedge (z \in y))$$

For example, if  $x = \{\{1, 2\}, \{3\}, \{4, 5\}\}$ , then  $u = \bigcup_x = \{1, 2, 3, 4, 5\}$  ( $y$  could, for example, be  $\{1, 2\}$ , in which case  $z$  would be 1 or 2).

We can define another structure related to set theory. This time define  $E = \bigcup_{n \geq 0} E_n$  by:

$$E_0 = \emptyset \quad \text{and} \quad E_{n+1} = \mathcal{P}(E_n)$$

where  $\mathcal{P}(X)$  denotes the power set of a set  $X$ . Thus:

- $E_0 = \emptyset$
- $E_1 = \{\emptyset\}$ , because the empty set is the only subset of  $E_0$ ,
- $E_2 = \{\emptyset, \{\emptyset\}\}$ , the subsets of  $E_1$ ,
- $E_3 = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \{\emptyset, \{\emptyset\}\}\}$ : the empty set, two singletons, and one pair,
- ...  $E_n$  contains a number of elements that grows very rapidly.

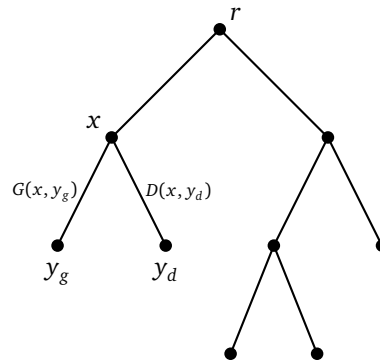
Membership here is ordinary membership; for example,  $\{\emptyset\} \in \{\emptyset, \{\emptyset\}\}$ , whereas  $\emptyset \notin \{\{\emptyset\}\}$ .

### Binary trees

We want to model the notion of a binary tree (each vertex has 0 or 2 children). Let  $\mathcal{L} = (r, G, D)$  be a language where  $r$  is a constant and  $G(x, y)$  and  $D(x, y)$  are binary predicates.

Define a structure  $\mathcal{S}$  for this language:

- $E$  is a set of vertices (possibly infinite),
- $r \in E$  is a distinguished vertex, called the *root*,
- $G \subset E^2$  is the *left-child* relation;  $G(x, y)$  is true if  $y$  is the left child of  $x$ ,
- $D \subset E^2$  is the *right-child* relation;  $D(x, y)$  is true if  $y$  is the right child of  $x$ .



Define a list of *axioms*, that is, formulas that must be true for every structure defining a binary tree.

- Every vertex has at most one left child and at most one right child:

$$\forall x \quad \forall y \quad \forall z \quad G(x, y) \wedge G(x, z) \rightarrow y = z$$

$$\forall x \quad \forall y \quad \forall z \quad D(x, y) \wedge D(x, z) \rightarrow y = z$$

- The left and right children are distinct:

$$\forall x \quad \forall y \quad \forall z \quad G(x, y) \wedge D(x, z) \rightarrow y \neq z$$

- A vertex has only one parent:

$$\forall x \quad \forall y \quad \forall z \quad G(x, z) \wedge G(y, z) \rightarrow x = y$$

$$\forall x \quad \forall y \quad \forall z \quad D(x, z) \wedge D(y, z) \rightarrow x = y$$

$$\forall x \quad \forall y \quad \forall z \quad G(x, z) \wedge D(y, z) \rightarrow x = y$$

- The root is nobody's child:

$$\forall x \quad \neg G(x, r)$$

$$\forall x \quad \neg D(x, r)$$

- Every vertex has either zero children or two children:

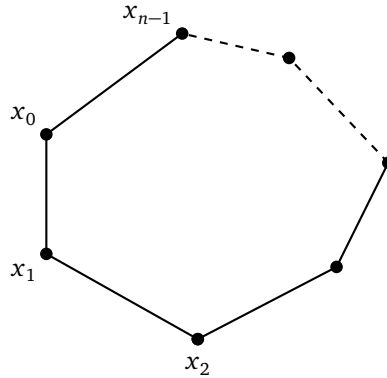
$$\forall x \quad (\exists y \quad G(x, y)) \leftrightarrow (\exists z \quad D(x, z))$$

- Optional axiom requiring the tree to be connected (in one piece): every vertex other than the root must have a parent:

$$\forall y \quad (y \neq r) \rightarrow (\exists x \quad G(x, y) \vee D(x, y))$$

(This ensures connectedness for finite trees.)

One subtle point remains to be expressed: a tree has no cycle (when following a path, one cannot return to its starting point).



**Cycle in a graph**

The absence of cycles cannot be expressed by a single formula, or even by finitely many formulas.

For every integer  $n \geq 1$  and every sequence of relations  $R_1, \dots, R_n \in \{G, D\}$ , we require that the corresponding path not return to its starting point. This is expressed by the formula:

$$\forall x_0 \quad \forall x_1 \quad \cdots \quad \forall x_{n-1} \quad \neg (R_1(x_0, x_1) \wedge R_2(x_1, x_2) \wedge \cdots \wedge R_n(x_{n-1}, x_0))$$

## Exercises

### Free variables, bound variables

**Exercise 1.** For each of the following formulas, indicate the scope of each quantifier and list the free and bound variables among  $x, y, z$ . Is the formula closed?

1.  $\forall x \quad P(x) \rightarrow Q(a, x)$
2.  $\exists z \quad Q(x, y) \wedge Q(y, z) \wedge Q(z, x)$
3.  $\forall x \quad (T(x, y, c) \rightarrow (\forall z \quad T(a, y, z)))$
4.  $(\forall x \quad P(x)) \wedge (\exists y \quad P(y)) \wedge (x \neq y)$
5.  $((\exists x \quad P(x)) \vee (\exists y \quad Q(x, y))) \rightarrow (\forall z \quad P(z))$
6.  $(\exists x \quad \exists y \quad P(x) \wedge P(y) \wedge (x \neq y)) \rightarrow P(z)$
7.  $\forall x \quad (Q(x, x) \rightarrow (\exists x \quad P(x)))$

**Exercise 2.** For each of the following formulas in a given structure, determine the free and bound variables (among  $x, y$ ). If the formula is closed, state whether the sentence is true or false. If the formula is not closed, state for which values of the free variables the formula becomes true.

1.  $\forall x \in \mathbb{N} \quad x^2 \geq y$
2.  $\forall x \in \mathbb{N} \quad \exists y \in \mathbb{N} \quad x \geq y + 1$
3.  $\forall y \in \mathbb{N} \quad \exists x \in \mathbb{N} \quad x \geq y + 1$
4.  $\exists y \in \mathbb{R} \quad x \neq y$
5.  $\forall y \in \mathbb{R} \quad x \neq y$
6.  $(\forall x \in \mathbb{R} \quad x^2 \geq 0) \rightarrow (y = x^2 \rightarrow y \geq 0)$

**Exercise 3.** Rewrite the formulas after each indicated substitution.

1.  $\mathcal{P}_1 : \forall x \quad (P(x) \wedge Q(x, y)) ; \mathcal{P}_1[b/y], \text{ then } \mathcal{P}_1[a/x], \text{ then } \mathcal{P}_1[x/y].$
2.  $\mathcal{P}_2 : P(x) \rightarrow (\exists y \quad Q(x, y)) ; \mathcal{P}_2[x + a/x], \text{ then } \mathcal{P}_2[y + b/y].$
3.  $\mathcal{P}_3 : \exists z \quad (x \neq y \rightarrow T(x, y, z)) ; \mathcal{P}_3[a/x, b/y, c/z].$
4.  $\mathcal{P}_4 : (\forall x \quad P(x)) \wedge (\exists y \quad Q(x, y)) \wedge (\forall z \quad \exists x \quad T(x, y, z)) ; \mathcal{P}_4[y/x, x/y, c/z].$

### Structures

**Exercise 4.** Consider the language  $\mathcal{L} = (a, f)$  with a constant  $a$  and a binary function  $f(x, y)$ . For each formula and each structure, give the interpretation of the formula and state whether this interpreted formula is true or false.

Formulas:

- $\mathcal{P}_1 : \forall x \quad \exists y \quad f(x, y) = a$
- $\mathcal{P}_2 : \forall x \quad \exists y \quad f(x, x) = f(f(y, y), y)$

- $\mathcal{P}_3 : \forall x \quad \forall y \quad f(x, y) = a \rightarrow x = a$

Structures:

1.  $\mathcal{S}_i : E = \mathbb{Z}, a = 0, f(x, y) = x + y$
2.  $\mathcal{S}_{ii} : E = \mathbb{R}, a = 0, f(x, y) = x - y$
3.  $\mathcal{S}_{iii} : E = ]0, +\infty[, a = 1, f(x, y) = xy$

**Exercise 5.** Consider the language  $\mathcal{L} = (a, b, P, Q)$ , where  $a, b$  are constants and  $P(x)$  and  $Q(x, y)$  are predicates (the variables are  $x, y$ ). For each formula and each structure, give the interpretation of the formula and state whether this interpreted formula is true or false.

Formulas:

- $\mathcal{P}_1 : \forall x \quad Q(x, a)$
- $\mathcal{P}_2 : \exists x \quad \forall y \quad P(x) \wedge Q(x, y)$
- $\mathcal{P}_3 : (\forall x \quad P(x)) \rightarrow (\exists y \quad Q(a, y))$

Structures:

1.  $\mathcal{S}_i : E = \mathbb{N}, a = 0, b = 1, P(x): "x \neq 1", Q(x, y): "x > y"$
2.  $\mathcal{S}_{ii} : E = M_2$ , the set of  $2 \times 2$  matrices with real entries;  $a$  is the zero matrix,  $b$  the identity matrix,  $P(x): "\det(x) \neq 0"$  (the determinant is nonzero),  $Q(x, y): "xy = I"$  (the product is the identity)

**Exercise 6.** Let  $\mathcal{L}$  be a language with a predicate  $R(x, y)$ . Consider  $E = \{r, s, t, u\}$ . For each of the following formulas, define an interpretation of the predicate  $R(x, y)$  in the form of an undirected graph so that the formula is true in this structure. (There may be several possible solutions.)

1.  $\forall x \quad \forall y \quad (x \neq y) \rightarrow R(x, y)$
2.  $(\forall x \quad R(x, x)) \wedge (\forall x \quad \exists y \quad \neg R(x, y))$
3.  $\forall x \quad \forall y \quad \forall z \quad (R(x, y) \wedge R(x, z)) \rightarrow (y = z)$
4.  $\neg (\forall x \quad \forall y \quad \forall z \quad (R(x, z) \wedge R(z, y)) \rightarrow R(x, y))$

# Satisfaction

*To assign a True/False value to a first-order logic formula, we must both define a mathematical structure and assign values to the variables. We will then be able to define logical consequence.*

## 1. True/false

In true/false logic, we assigned a value T/F to a formula by assigning each variable a value T/F. For a formula in a first-order language, things are a little more complicated. We must first define a structure in order to interpret the formula in that mathematical setting, and if there are free variables, we must also assign a mathematical value to each of them.

### 1.1. For all & There exists

The major innovation of first-order logic is, of course, the introduction of quantifiers. Consider a language  $\mathcal{L}$  and a predicate  $P(x)$ . Fix a structure  $\mathcal{S}$  for this language with domain  $E$ .

“ $\forall x P(x)$ ” is true iff for every  $x \in E$ ,  $P^{\mathcal{S}}(x)$  is true

“ $\exists x P(x)$ ” is true iff there is (at least) one  $x \in E$  such that  $P^{\mathcal{S}}(x)$  is true

Notice that we are obliged to define these notions using sentences in English, that is, using a metalanguage. The other important point is that this True/False evaluation has no meaning at the level of the language itself, but is carried out at the level of a structure. Recall that  $P^{\mathcal{S}}(x)$  is the interpretation of the predicate  $P(x)$  in the structure  $\mathcal{S}$ . Of course, the result may depend on the choice of structure.

#### Example.

- The formula

$$\exists x \quad x^2 < x$$

is false if we consider a structure with domain  $E_1 = [1, +\infty[$ , but true if the domain is  $\mathbb{R}$  (for example,  $(\frac{1}{2})^2 < \frac{1}{2}$ ).

- The formula

$$\forall x \quad (\exists y \quad y^2 = x)$$

is true if we consider the domain  $E_1 = [1, +\infty[$ , but false if the domain is  $\mathbb{R}$  (for example,  $-1$  is not the square of a real number).

Remark: if we have a formula  $\mathcal{P}$  whose only free variable is  $x$ , then we can also determine whether the sentences “ $\forall x \mathcal{P}$ ” and “ $\exists x \mathcal{P}$ ” are true or false in the structure. As usual, with a slight abuse of notation, we will often write  $\mathcal{P}(x)$  for a formula  $\mathcal{P}$  to indicate that it depends on the variable  $x$ .

## 1.2. True/False for closed formulas

Fix a language  $\mathcal{L}$ . Recall that a variable is *free* in a formula if one of its occurrences is not within the scope of a quantifier over that variable. A formula with no free variable is a *closed formula*.

### Definition.

Let  $\mathcal{S}$  be a structure and let  $Q$  be a closed formula. If the interpretation of  $Q$  in  $\mathcal{S}$  is true, then we say that  $Q$  is *satisfied* in  $\mathcal{S}$  and write:

$$\models_{\mathcal{S}} Q$$

We also say that  $\mathcal{S}$  is a *model* for  $Q$ . We say that  $Q$  is *satisfiable* if there is a structure  $\mathcal{S}$  such that  $\models_{\mathcal{S}} Q$ .

### Example.

Consider a suitable language and the formula:

$$Q: \quad \forall x \quad (x \times x = a) \rightarrow x = a$$

For the structure  $\mathcal{S}_1$  with domain  $E = \mathbb{R}$  and  $a^{\mathcal{S}_1} = 0$ , the formula is true, so indeed  $\models_{\mathcal{S}_1} Q$ .

On the other hand, for the structure  $\mathcal{S}_2$  with domain  $E = \mathbb{R}$ , but this time with  $a^{\mathcal{S}_2} = 1$ , the formula is false (indeed, if  $x^2 = 1$ , then  $x = +1$  or  $x = -1$ ).

## 1.3. Assignment

Consider a language  $\mathcal{L}$  and a structure  $\mathcal{S}$  with domain  $E$ . If  $x$  is a variable of  $\mathcal{L}$  and  $d$  an element of the set  $E$ , then  $x \mapsto d$  denotes the *assignment* of the value  $d$  to every free occurrence of  $x$  in the formulas of the language. We write  $Q[x \mapsto d]$  for the formula, interpreted in  $\mathcal{S}$ , in which the free occurrences of  $x$  are replaced by the value  $d$ .

For example, consider the formula  $Q: "x^2 \geq 7"$  for a structure with domain  $\mathbb{R}$ . Since the variable  $x$  is free, it makes no sense to ask whether  $Q$  is true or false. After assignment, however, the formula  $Q[x \mapsto 3]$  becomes " $3^2 \geq 7$ " and is therefore true, whereas  $Q[x \mapsto 2]$  becomes " $2^2 \geq 7$ " and is false.

More generally, an *assignment* is a function  $s: \text{Var} \rightarrow E$  that associates with each variable  $x_i$  of the language a value  $d_i$  in the domain:  $x_i \mapsto d_i$ . We write  $Q[s]$  for the application of an assignment  $s$  to a formula  $Q$ . Let us emphasize that only occurrences of free variables are assigned values.

## 1.4. Formulas with free variables

### Definition.

Let  $Q$  be a formula. We say that  $Q$  is *satisfied* in  $\mathcal{S}$  under the assignment  $s$  if the interpretation  $Q[s]$  is true. We then write  $\models_{\mathcal{S}} Q[s]$ , or also:

$$\models_{\mathcal{S},s} Q$$

Of course, the assignment matters only if there are free variables. Assignment is the counterpart of evaluation in true/false logic (where we replaced each propositional variable by a value T/F): here it is more general because  $x$  is replaced by any value from the set  $E$  (the domain).

### Example.

Consider a language in which we can write the formula:

$$Q: \quad \forall x \quad y \leq x$$

The variable  $y$  is free. We will consider a structure  $\mathcal{S}_1$  with domain  $\mathbb{N}$  and a structure  $\mathcal{S}_2$  with domain  $\mathbb{R}$  (and the usual order relation). It makes no sense to ask whether  $\mathcal{Q}$  is true, either in  $\mathcal{S}_1$  or in  $\mathcal{S}_2$ .

- In  $\mathcal{S}_1$ , consider the assignment  $s : y \mapsto 0$ . We indeed have

$$\mathcal{Q}[y \mapsto 0] : \quad \forall x \in \mathbb{N} \quad 0 \leq x$$

and so the statement  $\mathcal{Q}[y \mapsto 0]$  is true; hence  $\models_{\mathcal{S}_1, y \mapsto 0} \mathcal{Q}$ .

- In  $\mathcal{S}_2$ , the assignment  $s : y \mapsto 0$  gives:

$$\mathcal{Q}[y \mapsto 0] : \quad \forall x \in \mathbb{R} \quad 0 \leq x$$

which is, of course, false.

- The assignment  $y \mapsto 1$  gives false statements for both structures.
- Finally, let  $s$  be the assignment such that  $x \mapsto 5$  and  $y \mapsto 1$ . Since  $x$  is not a free variable,  $s$  does not assign values to the occurrences of  $x$  in the formula, which becomes  $\mathcal{Q}[s]$ : “ $\forall x \quad 1 \leq x$ ”.

## Validity

### Definition.

Let  $\mathcal{Q}$  be a formula of a language  $\mathcal{L}$ . We say that  $\mathcal{Q}$  is a **validity** if, for every structure  $\mathcal{S}$  and every assignment  $s$ , the interpretation of  $\mathcal{Q}[s]$  in  $\mathcal{S}$  is true. We then write:

$$\models \mathcal{Q}$$

Thus, this means that  $\models_{\mathcal{S}, s} \mathcal{Q}$  for every  $\mathcal{S}$  and  $s$ .

A validity is the analogue of a tautology in propositional logic. If  $\models \neg \mathcal{Q}$ , this means that  $\mathcal{Q}$  is always false (for every  $\mathcal{S}$  and  $s$ ); such a formula  $\mathcal{Q}$  is called a **contradiction** or is said to be **unsatisfiable**.

### Example.

- “ $\forall x \quad x = x$ ” is a validity.
- “ $x = x$ ” is also a validity.
- “ $\forall x \quad P(x) \wedge Q(x) \rightarrow P(x)$ ” is a validity (we do not need to know how the predicates  $P$  and  $Q$  will be interpreted).
- “ $\exists x \exists y \quad x \neq y$ ” is not a validity. It is very often true, but not if we choose a structure whose domain has only one element, for example  $E = \{0\}$ , in which there cannot be two distinct elements.
- “ $x < x$ ” is not a validity (it is false over  $\mathbb{R}$  if we interpret “ $<$ ” as strict inequality, but true if we interpret it as non-strict inequality).
- “ $\exists x \quad P(x) \wedge \neg P(x)$ ” is a contradiction. Even if we do not know how the predicate  $P$  will be interpreted, we can never have both  $P(x)$  and its negation at the same time. (Note that the domain of a structure is nonempty; this is important here.)

## 2. Logical consequence

### 2.1. Definition

We generalize the notion of satisfaction as follows: if the assumptions  $\mathcal{P}_1, \dots, \mathcal{P}_l$  are satisfied, then the conclusion  $\mathcal{Q}$  is also satisfied.

**Definition.**

We say that  $\mathcal{Q}$  is a **logical consequence** of  $\mathcal{P}_1, \dots, \mathcal{P}_l$  if, for every structure  $\mathcal{S}$  and every assignment  $s$  that make  $\mathcal{P}_1, \dots, \mathcal{P}_l$  simultaneously true,  $\mathcal{Q}$  is also true.

We then write:

$$\mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q}$$

Similarly, if  $\Gamma$  is a set of formulas, we write  $\Gamma \models \mathcal{Q}$ .

Here is an important example:

$$\forall x \mathcal{P}(x) \models \exists x \mathcal{P}(x)$$

If a formula is true for every element, then it is true for at least one element. Note once again that it is important that we required the domains of structures always to be nonempty.

There are natural logical consequences inherited from propositional logic:

- $\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$
- $\mathcal{P}, \mathcal{Q} \models \mathcal{P} \wedge \mathcal{Q}$
- $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \models \mathcal{Q}$
- if  $\mathcal{P} \models \mathcal{Q}$  and  $\mathcal{Q} \models \mathcal{R}$ , then  $\mathcal{P} \models \mathcal{R}$

### 2.2. Logical equivalence

**Definition.**

If  $\mathcal{P} \models \mathcal{Q}$  and  $\mathcal{Q} \models \mathcal{P}$ , we say that  $\mathcal{P}$  and  $\mathcal{Q}$  are **logically equivalent** and write:

$$\mathcal{P} \equiv \mathcal{Q}$$

When this is the case, it means that  $\mathcal{P}$  and  $\mathcal{Q}$  are satisfied at the same time. Another way to define logical equivalence would be  $\models (\mathcal{P} \leftrightarrow \mathcal{Q})$ .

Of course, we recover equivalences from propositional logic, for example:

- $\neg(\mathcal{P} \vee \mathcal{Q}) \equiv \neg\mathcal{P} \wedge \neg\mathcal{Q}$  (De Morgan's law),
- $\mathcal{P} \rightarrow \mathcal{Q} \equiv \neg\mathcal{P} \vee \mathcal{Q}$  (definition of implication),
- etc.

## 2.3. Equivalences involving quantifiers

Let us examine logical equivalences involving quantifiers.

**Negation of quantifiers.**

$$\neg(\forall x \mathcal{P}(x)) \equiv \exists x \neg \mathcal{P}(x)$$

$$\neg(\exists x \mathcal{P}(x)) \equiv \forall x \neg \mathcal{P}(x)$$

Roughly speaking, remember that the negation of “for every” is “there exists”, and conversely. These equivalences imply that if we have a formula  $\mathcal{Q}$  containing quantifiers, then we can always find an equivalent formula containing only the quantifier “for every” (or only the quantifier “there exists”).

**Example.**

$$\begin{aligned} & \neg(\forall x \ P(x) \wedge Q(x)) \\ \equiv & \exists x \ \neg(P(x) \wedge Q(x)) \\ \equiv & \exists x \ (\neg P(x)) \vee (\neg Q(x)) \end{aligned}$$

$$\begin{aligned} & \neg(\exists x \ P(x) \rightarrow Q(x)) \\ \equiv & \forall x \ \neg(P(x) \rightarrow Q(x)) \\ \equiv & \forall x \ \neg((\neg P(x)) \vee Q(x)) \\ \equiv & \forall x \ P(x) \wedge (\neg Q(x)) \end{aligned}$$

**Order of quantifiers.**

We can interchange two consecutive “for every” quantifiers, or two consecutive “there exists” quantifiers:

$$\forall x \forall y \mathcal{P}(x, y) \equiv \forall y \forall x \mathcal{P}(x, y)$$

$$\exists x \exists y \mathcal{P}(x, y) \equiv \exists y \exists x \mathcal{P}(x, y)$$

On the other hand, in general we cannot interchange a “for every” and a “there exists”:

$$\forall x \exists y \mathcal{P}(x, y) \text{ is not equivalent to } \exists y \forall x \mathcal{P}(x, y)$$

For example:

$$\forall x \in \mathbb{R} \ \exists y \in \mathbb{R} \quad x + y = 0 \quad \text{is true,}$$

whereas

$$\exists y \in \mathbb{R} \ \forall x \in \mathbb{R} \quad x + y = 0 \quad \text{is false.}$$

Everyday examples can also make this clear: “every student has a phone number,” but it is not true that “there is one phone number shared by all students.”

**Other useful equivalences.**

$$\begin{aligned}
\forall x \ (P(x) \wedge Q(x)) &\equiv (\forall x \ P(x)) \wedge (\forall x \ Q(x)) \\
\exists x \ (P(x) \vee Q(x)) &\equiv (\exists x \ P(x)) \vee (\exists x \ Q(x)) \\
\forall x \ (P(x) \rightarrow Q) &\equiv (\exists x \ P(x)) \rightarrow Q \quad \text{if } x \text{ does not occur in } Q
\end{aligned}$$

Finally, we can rename a variable  $x$  as a new variable  $y$  (which does not occur in  $P(x)$ ):

$$\forall x \ P(x) \equiv \forall y \ P[y/x]$$

## 2.4. Satisfaction of quantifiers

Let us define the satisfaction of quantifiers a little more formally:

$$\models_{S,s} \forall x \ P(x) \quad \text{iff} \quad \text{for every } d \in E, \models_{S,s[x \mapsto d]} P(x)$$

The notation  $s[x \mapsto d]$  denotes the assignment that maps  $x$  to  $d$  and agrees with  $s$  on all other variables.

$$\models_{S,s} \exists x \ P(x) \quad \text{iff} \quad \text{there exists } d \in E \text{ such that } \models_{S,s[x \mapsto d]} P(x)$$

## 2.5. Assignment vs. substitution

This section is more technical and may be skipped on a first reading. The important point is that assignment and substitution are two different operations, but they are compatible.

Consider a language  $\mathcal{L}$  and a structure  $S$  with domain  $E$ . Recall:

- the **substitution**  $P[t/x]$  replaces every free occurrence of  $x$  by the term  $t$ . We start with a formula  $P$  of the language  $\mathcal{L}$  and obtain a formula  $P[t/x]$  that is still in the same language.
- the **assignment**  $P[x \mapsto d]$  replaces every free occurrence of  $x$  by the value  $d \in E$ . The formula  $P$  is still a formula of the language  $\mathcal{L}$ , but  $P[x \mapsto d]$  is a formula interpreted in the structure  $S$ .

Once we have clearly understood the difference, it is reassuring to know that combining substitution followed by assignment is equivalent to combining them in the other order:

$$\models_{S,s} P[t/x] \quad \text{iff} \quad \models_{S,s[x \mapsto t^{S,s}]} P$$

### Example.

Let  $\mathcal{L} = (0, 1, \leq, +)$  be a language. Write  $2 = 1 + 1$  and  $3 = 2 + 1$ . Consider the formula:

$$P(x): \quad \forall y \quad x \leq y + 3$$

Let the term be  $t = x + 1$ .

Consider the structure  $S$  with domain  $E = \mathbb{N}$  naturally associated with  $\mathcal{L}$ .

- The substitution  $P[t/x]$  gives the following formula, which remains in the language  $\mathcal{L}$ :

$$P[t/x]: \quad \forall y \quad x + 1 \leq y + 3$$

- Start again from  $P$ . The assignment  $s : x \mapsto 1$  transforms a formula of  $\mathcal{L}$  into a formula interpreted in the structure:

$$P[x \mapsto 1]: \quad \forall y \in \mathbb{N} \quad 1 \leq y + 3$$

- Substitution followed by assignment. Start from  $P$ , substitute  $t$  for  $x$  to obtain  $P[t/x]$  as above,

and then assign 1 to the variable  $x$ . We then obtain:

$$\mathcal{P}[t/x][x \mapsto 1]: \quad \forall y \in \mathbb{N} \quad 2 \leq y + 3$$

- Assignment first. Start with the term  $t = x + 1$  and assign the value 1 to  $x$  in this term. We obtain the term  $t^{S,s} = 2 \in E$ . Now, in  $\mathcal{P}$ , replace every free occurrence of  $x$  by  $t^{S,s}$ . We obtain the same formula as before:

$$\mathcal{P}[x \mapsto 2]: \quad \forall y \in \mathbb{N} \quad 2 \leq y + 3$$

## 2.6. Summary of the concepts

| Notation                                        | Terminology                          | Description                                                                                                                                               |
|-------------------------------------------------|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\models_S Q$                                   | Satisfaction (closed formula). Model | The closed formula $Q$ is true in the structure $S$ . We say that $S$ is a model of $Q$ .                                                                 |
| $\models_{S,s} Q$                               | Satisfaction under an assignment     | The formula $Q$ is true in $S$ under the assignment $s$ .                                                                                                 |
|                                                 | Satisfiable formula                  | Formula true in at least one structure (and under an assignment if necessary).                                                                            |
| $\models Q$                                     | Validity                             | Formula $Q$ true in every structure and under every assignment.                                                                                           |
| $\models \neg Q$                                | Contradiction                        | Formula $Q$ always false, in every structure and under every assignment.                                                                                  |
| $\mathcal{P}_1, \dots, \mathcal{P}_l \models Q$ | Logical consequence                  | Every structure and assignment that simultaneously satisfy the $\mathcal{P}_i$ also satisfy $Q$ .                                                         |
| $\mathcal{P} \equiv Q$                          | Logical equivalence                  | $\mathcal{P} \models Q$ and $Q \models \mathcal{P}$ . For every structure $S$ and every assignment $s$ , $\mathcal{P}$ and $Q$ have the same truth value. |

## Exercises

### True/false

**Exercise 1.** For each of the following closed formulas, determine whether it is true or false when interpreted in the different structures; that is, does  $\models_S Q$  hold? Consider  $\mathcal{L} = (a, f(x, y))$  and the formulas:

$$Q_1: \quad \forall x \exists y \quad f(x, y) = a$$

$$Q_2: \quad \forall x \forall y \quad f(x, y) = a \rightarrow (x = a) \vee (y = a)$$

$$Q_3: \quad \forall x \quad (x \neq a) \rightarrow (\exists y \quad f(x, y) \neq a)$$

Here are the structures:

$$\mathcal{S}_i : \mathbb{N}, a = 0, f(x, y) = x + y$$

$$\mathcal{S}_{ii} : \mathbb{N}, a = 0, f(x, y) = xy$$

$$\mathcal{S}_{iii} : \mathbb{N}, a = 1, f(x, y) = xy$$

**Exercise 2.** Consider the language  $\mathcal{L} = (<, f(x))$  and the structure  $\mathcal{S}$  with domain  $\mathbb{R}$  and  $f(x) = e^x$ . For the following formulas and given assignments, determine whether the interpretation is true or false; that is, does  $\models_{\mathcal{S}, s} \mathcal{Q}$  hold?

$$1. \mathcal{Q}_1 : \quad \forall x \quad y < f(x) \quad s_1 : y \mapsto 0; \text{ then } s_2 : y \mapsto e$$

$$2. \mathcal{Q}_2 : \quad \exists x \quad x < f(y) \quad s_1 : y \mapsto 0; \text{ then } s_2 : y \mapsto e$$

$$3. \mathcal{Q}_3 : \quad \neg(f(x) < f(y)) \quad s : x \mapsto 0, y \mapsto e$$

**Exercise 3.** Consider the language  $\mathcal{L} = (0, 1, <, +, \times)$  and the naturally associated structure  $\mathcal{S}$  with domain  $\mathbb{R}$ . For each of the following formulas, find all assignments  $s$  to the free variables that make the formula  $\mathcal{Q}$  true in the structure:

$$1. \mathcal{Q}_1 : \quad \forall x \quad (x \neq 0) \rightarrow x \times y \neq 0$$

$$2. \mathcal{Q}_2 : \quad \exists x \quad x \times x = y$$

$$3. \mathcal{Q}_3 : \quad \exists y \quad x < y \times y$$

For each of the following formulas, represent in the plane  $\mathbb{R}^2$  the set of values that, when assigned to  $(x, y)$ , make the formula  $\mathcal{Q}$  true:

$$4. \mathcal{Q}_4 : \quad x \times x < y$$

$$5. \mathcal{Q}_5 : \quad (\exists x \quad x + y = 0) \vee (\forall y \quad x \times y = 0)$$

$$6. \mathcal{Q}_6 : \quad (x \times x = y \times y) \rightarrow (x \neq y)$$

**Exercise 4.** Are the following formulas validities (true for every  $\mathcal{S}$  and  $s$ ), contradictions (false for every  $\mathcal{S}$  and  $s$ ), or neither (in which case justify your answer)?

$$1. (\forall x P(x)) \rightarrow (\exists x P(x))$$

$$2. (\exists x P(x)) \rightarrow (\forall x P(x))$$

$$3. \forall x \exists y \quad (x = y) \wedge (x \neq y)$$

$$4. \forall x \quad Q(x, y) \rightarrow Q(y, x)$$

$$5. x = y$$

$$6. \exists y \quad (x = y) \vee (x \neq y)$$

## Logical consequence

**Exercise 5.** Determine whether each of the following formulas  $\mathcal{Q}$  is a logical consequence of  $\mathcal{P}_1, \dots, \mathcal{P}_l$ ; that is, does  $\mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q}$  hold?

$$1. \mathcal{P} : \forall x \quad x < y, \quad \mathcal{Q} : \exists x \quad x < y$$

$$2. \mathcal{P} : \neg(x = y), \quad \mathcal{Q} : x < y$$

$$3. \mathcal{P}_1 : \forall x \quad P(x), \quad \mathcal{P}_2 : \exists y \quad Q(y), \quad \mathcal{Q} : \exists x \quad P(x) \wedge Q(x)$$

$$4. \mathcal{P}_1 : \exists x \quad P(x), \quad \mathcal{P}_2 : \exists x \quad Q(x), \quad \mathcal{Q} : \exists x \quad P(x) \wedge Q(x)$$

**Exercise 6.** Write a sentence in English that explains or justifies each of the following properties:

1.  $\mathcal{P} \models \mathcal{P}$
2.  $\mathcal{P} \models \mathcal{Q}$  iff  $\models (\mathcal{P} \rightarrow \mathcal{Q})$
3. if  $\mathcal{P} \models \mathcal{Q}$  then  $\mathcal{P}, \mathcal{R} \models \mathcal{Q}$
4. if  $\mathcal{P} \models \mathcal{Q}$  and  $\mathcal{Q} \models \mathcal{R}$  then  $\mathcal{P} \models \mathcal{R}$

**Exercise 7.** For each of the following formulas (given their natural interpretation), write the negation of the statement and determine whether the original statement is true or false.

1.  $\exists x \in \mathbb{R} \quad x^2 = 2$
2.  $\forall x \in \mathbb{N} \quad x > 2$
3.  $\exists x \in ]0, +\infty[ \quad \forall y \in ]0, +\infty[ \quad x + y = 1$
4.  $\forall z \in \mathbb{C} \quad z \neq 0 \rightarrow z^2 \neq 0$
5.  $\forall x \in \mathbb{R} \quad \forall y \in \mathbb{R} \quad xy = 0 \rightarrow (x = 0) \vee (y = 0)$
6.  $\forall x \in \mathbb{R} \quad x > 1 \rightarrow (\exists y \in \mathbb{R} \quad (y > 1) \wedge (x > y))$

**Exercise 8.** For each pair of formulas  $\mathcal{P}$ ,  $\mathcal{Q}$ , determine whether we have the logical consequence  $\mathcal{P} \models \mathcal{Q}$ , or  $\mathcal{Q} \models \mathcal{P}$ , or the logical equivalence  $\mathcal{P} \equiv \mathcal{Q}$ , or none of these.

1.  $\mathcal{A} \vee \mathcal{B}$  and  $\mathcal{B} \vee \mathcal{A}$
2.  $\neg(\mathcal{A} \vee \mathcal{B})$  and  $(\neg\mathcal{A}) \vee (\neg\mathcal{B})$
3.  $\exists x \neg\mathcal{A}$  and  $\forall x \mathcal{A}$
4.  $\forall x \mathcal{A} \rightarrow \mathcal{B}$  and  $\exists x \mathcal{B} \wedge (\neg\mathcal{A})$
5.  $\forall x \mathcal{A}$  and  $\mathcal{A}$
6.  $\exists x \mathcal{A}$  and  $\mathcal{A}$

**Exercise 9.** Show that the following are **not** logical equivalences:

1.  $\exists x \mathcal{P} \wedge \mathcal{Q}$  and  $(\exists x \mathcal{P}) \wedge (\exists x \mathcal{Q})$
2.  $\forall x \mathcal{P} \vee \mathcal{Q}$  and  $(\forall x \mathcal{P}) \wedge (\forall x \mathcal{Q})$
3.  $\forall x \exists y \mathcal{P}$  and  $\exists y \forall x \mathcal{P}$
4.  $\forall x (\mathcal{P} \rightarrow \mathcal{Q})$  and  $(\exists x \mathcal{P}) \rightarrow \mathcal{Q}$

**Exercise 10.** Let  $\mathcal{L}$  be a language. An *axiom* is a closed formula  $\mathcal{A}$  that we require to be true in the structures under consideration; that is, we consider only structures  $\mathcal{S}$  such that  $\models_{\mathcal{S}} \mathcal{A}$ .

1. What can we say about a structure  $\mathcal{S}$  that satisfies the axiom

$$\mathcal{A}: \quad \forall x \forall y \quad x = y$$

2. Same question for:

$$\mathcal{A}: \quad \exists x \exists y \quad (x \neq y) \wedge (\forall z \quad (z = x) \vee (z = y))$$

*We adapt the construction of proofs to the quantifiers of first-order logic.*

## 1. Proofs in first-order logic

### 1.1. Review

Recall that a proof consists in starting from a set of formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  and arriving at a formula  $\mathcal{Q}$  in a finite number of steps by following precise rules. If we have obtained a proof, we write:

$$\mathcal{P}_1, \dots, \mathcal{P}_l \vdash \mathcal{Q}$$

For first-order logic, we first retain all the rules of inference for proofs in true/false logic. We refer to the chapter “Proofs” in Part One. We will add new rules involving quantifiers. Recall that there is still no connection between what we prove and what is true. This connection will be established in the following chapters. In particular, in the next chapter we will show that everything that is proved is true. In this chapter, we fix a language  $\mathcal{L}$ . A proof involves only this language. There are no structures or assignments to consider. Thus, the formulas in a proof remain in  $\mathcal{L}$ , which is one respect in which the notion of proof ( $\vdash$ ) is simpler than the notion of satisfaction ( $\models$ ).

### 1.2. Rules for quantifiers

We present four new rules involving quantifiers: the introduction and elimination rules for  $\forall$  and  $\exists$ . An additional rule concerns equality. We present each rule briefly before returning to it in detail in the next section. There are two easy rules and two difficult rules. The difficulty lies not in the boxed rule, but rather in its conditions of application.

To clarify the notation for the rules, when we have a formula  $\mathcal{P}$  and are interested in its dependence on the variable  $x$ , we naturally denote it by  $\mathcal{P}(x)$ . Note that it may also depend on other variables. For a formula  $\mathcal{P}(x)$ , we will write  $\mathcal{P}(t)$  instead of  $\mathcal{P}[t/x]$ , the substitution of the term  $t$  for the variable  $x$ .

**Rule I.  $\forall$  elimination.**

|                                                  |
|--------------------------------------------------|
| $\forall x \mathcal{P}(x) \vdash \mathcal{P}(c)$ |
|--------------------------------------------------|

- *Conditions.*  $c$  may be any constant in the language;  $c$  may also be a *closed term*, that is, a term with no free variable.
- *Remark.* Thus, under the assumption  $\forall x \mathcal{P}(x)$ , we can replace every free occurrence of  $x$  by  $c$ , and this proves  $\mathcal{P}(c)$  (which is in fact  $\mathcal{P}[c/x]$ ). Here it does not matter if the constant (or term)  $c$  has already appeared earlier in the proof.

- *Example.* For any constant  $c$ :

$$\forall x \exists y P(x) \wedge Q(x, y) \vdash \exists y P(c) \wedge Q(c, y)$$

**Rule II.  $\forall$  introduction.**

$$\boxed{\mathcal{P}(c) \vdash \forall x \mathcal{P}(x)}$$

- *Conditions.*  $c$  denotes a constant of  $\mathcal{L}$ , but it must be *arbitrary*, that is, it must not appear in any assumption currently in use.
- *Remark.* This is a tricky rule because here  $c$  does not really play the role of a constant, but rather that of a *free variable*. It is best to use a name  $c$  that does not appear at all in the preceding lines of the proof.
- *Example.* Let  $c$  be arbitrary:

$$c^2 \geq 0 \vdash \forall x x^2 \geq 0$$

Indeed, if we have proved  $c^2 \geq 0$  for an arbitrary  $c$ , then  $x^2 \geq 0$  for every  $x$ .

**Rule III.  $\exists$  introduction.**

$$\boxed{\mathcal{P}(c) \vdash \exists x \mathcal{P}(x)}$$

- *Conditions.*  $c$  is a constant in the language or a term with no free variable.
- *Remark.*  $c$  is called a *witness* for the quantifier  $\exists$ . The symbol  $c$  may have appeared earlier in the proof.
- *Example.*

$$c^2 \neq c \vdash \exists x x^2 \neq x$$

**Rule IV.  $\exists$  elimination.**

$$\boxed{\exists x \mathcal{P}(x), [\mathcal{P}(c) \vdash \mathcal{Q}] \vdash \mathcal{Q}}$$

The notation in brackets corresponds to a subproof. This is clearer when written in tabular form:

$$\begin{array}{c|c|c} & \exists x \mathcal{P}(x) & \\ \hline & \mathcal{P}(c) & \text{Temporary assumption} \\ & \vdots & \\ & \mathcal{Q} & \\ \hline \vdash & \mathcal{Q} & \end{array}$$

- *Conditions.*  $c$  must be arbitrary: it must appear neither in  $\mathcal{Q}$  nor in  $\exists x \mathcal{P}(x)$ , nor in any assumption currently in use. It is called a *fresh constant* (or, by abuse of terminology, a *fresh variable*).
- *Remark.* The condition is not obvious at first, but the principle is natural: if we know that there is an element satisfying  $\mathcal{P}$ , and assuming  $\mathcal{P}(c)$  for an arbitrary  $c$  leads to  $\mathcal{Q}$ , then the conclusion  $\mathcal{Q}$  is proved.

- *Example.*

$$\exists x \, f(x) = 0, \forall x \, (f(x) = 0 \rightarrow g(x) = 1) \vdash \exists x \, g(x) = 1$$

This is an almost direct application of Rule IV; see the next section for details.

### Rule V. Equality.

$$t = s, \mathcal{P}(t) \vdash \mathcal{P}(s)$$

- *Conditions.*  $t$  and  $s$  are terms in the language. The substitutions of  $t$  or  $s$  must of course be legitimate (with no capture).
- *Remark.* There is no trap with equality. Two equal things are considered identical and therefore freely interchangeable.
- *Example.*

$$a = x + 1, a > 7 \vdash x + 1 > 7$$

## 1.3. Examples of proofs

### Example.

$$\forall x \, \mathcal{P}(x) \vdash \exists x \, \mathcal{P}(x)$$

|          |                               |                        |
|----------|-------------------------------|------------------------|
|          | $\forall x \, \mathcal{P}(x)$ | Assumption             |
|          | $\mathcal{P}(c)$              | $\forall$ elimination  |
| $\vdash$ | $\exists x \, \mathcal{P}(x)$ | $\exists$ introduction |

The proof rules implicitly assume that we are working with nonempty sets. This is consistent with the domain of a structure, which must be a nonempty set.

### Example.

Let  $\mathcal{P}$  be a formula in which the variable  $x$  does not appear.

$$\forall x \, (\mathcal{P} \rightarrow \mathcal{Q}(x)) \vdash \mathcal{P} \rightarrow (\forall x \, \mathcal{Q}(x))$$

|          |                                                         |                                                      |
|----------|---------------------------------------------------------|------------------------------------------------------|
| 1        | $\forall x \, (\mathcal{P} \rightarrow \mathcal{Q}(x))$ |                                                      |
| 2        | $\mathcal{P}$                                           | Temporary assumption                                 |
| 3        | $\mathcal{P} \rightarrow \mathcal{Q}(c)$                | $\forall$ elimination from 1                         |
| 4        | $\mathcal{Q}(c)$                                        |                                                      |
| 5        | $\forall x \, \mathcal{Q}(x)$                           | $\forall$ introduction; $c$ appears in no assumption |
| $\vdash$ | $\mathcal{P} \rightarrow (\forall x \, \mathcal{Q}(x))$ |                                                      |

**Example.**

$$\forall x (\mathcal{P}(x) \rightarrow \mathcal{Q}(x)), \exists x \mathcal{P}(x) \vdash \exists x \mathcal{Q}(x)$$

This is quite natural: if  $\mathcal{P}(x)$  always implies  $\mathcal{Q}(x)$  and one of the  $\mathcal{P}(x)$  is true, then one of the  $\mathcal{Q}(x)$  is true. This is a variant of *modus ponens* in propositional logic:  $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}$ .

|          |                                                         |                                      |
|----------|---------------------------------------------------------|--------------------------------------|
| 1        | $\forall x (\mathcal{P}(x) \rightarrow \mathcal{Q}(x))$ |                                      |
| 2        | $\exists x \mathcal{P}(x)$                              |                                      |
| 3        | $\mathcal{P}(c)$                                        | Temporary assumption                 |
| 4        | $\mathcal{P}(c) \rightarrow \mathcal{Q}(c)$             | $\forall$ elimination from 1         |
| 5        | $\mathcal{Q}(c)$                                        |                                      |
| 6        | $\exists x \mathcal{Q}(x)$                              | $\exists$ introduction               |
| $\vdash$ | $\exists x \mathcal{Q}(x)$                              | $\exists$ elimination from 2 via 3–6 |

**Example.**

$$\exists x \forall y \mathcal{P}(x, y) \vdash \forall y \exists x \mathcal{P}(x, y)$$

We can therefore switch an  $\exists \forall$  to a  $\forall \exists$  (but not the other way around).

|          |                                           |                                      |
|----------|-------------------------------------------|--------------------------------------|
| 1        | $\exists x \forall y \mathcal{P}(x, y)$   |                                      |
| 2        | $\forall y \mathcal{P}(a, y)$             | With $a$ arbitrary                   |
| 3        | $\mathcal{P}(a, b)$                       | With $b$ arbitrary                   |
| 4        | $\exists x \mathcal{P}(x, b)$             | $\exists$ introduction               |
| 5        | $\forall y (\exists x \mathcal{P}(x, y))$ | $\forall$ introduction               |
| $\vdash$ | $\forall y \exists x \mathcal{P}(x, y)$   | $\exists$ elimination from 1 via 2–5 |

## 2. Negation of quantifiers and derived rules

### 2.1. Negation of quantifiers

**Derived rule. Negation of quantifiers.**

$$\neg (\forall x \mathcal{P}(x)) \vdash \exists x (\neg \mathcal{P}(x))$$

$$\neg (\exists x \mathcal{P}(x)) \vdash \forall x (\neg \mathcal{P}(x))$$

We are not surprised: the negation of a “for every” is a “there exists”, and conversely.

Examples:

1.  $\neg (\forall x \ f(x) = g(x)) \vdash \exists x \ (f(x) \neq g(x))$
2.  $\neg (\exists x \ P(x) \wedge Q(x)) \vdash \forall x \ (\neg P(x)) \vee (\neg Q(x))$
3.  $\neg (\forall x \ P(x) \rightarrow Q(x)) \vdash \exists x \ P(x) \wedge (\neg Q(x))$

## 2.2. Other derived rules

**Derived rule. Renaming.**

$$\forall x \ P(x) \vdash \forall y \ P(y) \quad \text{and} \quad \exists x \ P(x) \vdash \exists y \ P(y)$$

The variable  $y$  must not occur as a free variable in the formula  $\forall x \ P(x)$  or in  $\exists x \ P(x)$ .

**Derived rule. Switching two  $\forall$  quantifiers or two  $\exists$  quantifiers.**

$$\forall x \ \forall y \ P(x, y) \vdash \forall y \ \forall x \ P(x, y)$$

$$\exists x \ \exists y \ P(x, y) \vdash \exists y \ \exists x \ P(x, y)$$

**Derived rule. Switching  $\exists \forall$  to  $\forall \exists$ .**

$$\exists x \ \forall y \ P(x, y) \vdash \forall y \ \exists x \ P(x, y)$$

We proved it in an earlier example.

**Derived rule. Interchanging  $\forall$  and  $\wedge$ .**

$$\forall x \ (P(x) \wedge Q(x)) \vdash (\forall x \ P(x)) \wedge (\forall x \ Q(x))$$

$$(\forall x \ P(x)) \wedge (\forall x \ Q(x)) \vdash \forall x \ (P(x) \wedge Q(x))$$

Thus, we can go in either direction. The formulas are interderivable.

**Derived rule. Interchanging  $\exists$  and  $\vee$ .**

$$\exists x \ (P(x) \vee Q(x)) \vdash (\exists x \ P(x)) \vee (\exists x \ Q(x))$$

$$(\exists x \ P(x)) \vee (\exists x \ Q(x)) \vdash \exists x \ (P(x) \vee Q(x))$$

## 2.3. Proofs of the derived rules

Let us look at a few proofs of the derived rules.

### Example.

The renaming rules are natural. The proofs are simple notational manipulations.

$$\forall x \ P(x) \vdash \forall y \ P(y)$$

|          |                            |                                      |
|----------|----------------------------|--------------------------------------|
|          | $\forall x \mathcal{P}(x)$ |                                      |
|          | $\mathcal{P}(c)$           | $\forall$ elimination, $c$ arbitrary |
| $\vdash$ | $\forall y \mathcal{P}(y)$ | $\forall$ introduction               |

$$\exists x \mathcal{P}(x) \vdash \exists y \mathcal{P}(y)$$

|          |                            |                                      |
|----------|----------------------------|--------------------------------------|
|          | $\exists x \mathcal{P}(x)$ |                                      |
|          | $\mathcal{P}(c)$           | $c$ a constant                       |
|          | $\exists y \mathcal{P}(y)$ | $\exists$ introduction               |
| $\vdash$ | $\exists y \mathcal{P}(y)$ | Elimination of the initial $\exists$ |

**Example.**

$$\forall x \forall y \mathcal{P}(x, y) \vdash \forall y \forall x \mathcal{P}(x, y)$$

|          |                                         |                                                     |
|----------|-----------------------------------------|-----------------------------------------------------|
|          | $\forall x \forall y \mathcal{P}(x, y)$ |                                                     |
|          | $\forall y \mathcal{P}(a, y)$           | Elimination of the first $\forall$ , $a$ arbitrary  |
|          | $\mathcal{P}(a, b)$                     | Elimination of the second $\forall$ , $b$ arbitrary |
|          | $\forall x \mathcal{P}(x, b)$           | Introduction of a $\forall$                         |
| $\vdash$ | $\forall y \forall x \mathcal{P}(x, y)$ | Introduction of another $\forall$                   |

**Example.**

$$(\forall x \mathcal{P}(x)) \wedge (\forall x \mathcal{Q}(x)) \vdash \forall x (\mathcal{P}(x) \wedge \mathcal{Q}(x))$$

Here is the proof; it is up to you to state the rules used explicitly.

|          |                                                                |  |
|----------|----------------------------------------------------------------|--|
|          | $(\forall x \mathcal{P}(x)) \wedge (\forall x \mathcal{Q}(x))$ |  |
|          | $\forall x \mathcal{P}(x)$                                     |  |
|          | $\mathcal{P}(c)$                                               |  |
|          | $\forall x \mathcal{Q}(x)$                                     |  |
|          | $\mathcal{Q}(c)$                                               |  |
|          | $\mathcal{P}(c) \wedge \mathcal{Q}(c)$                         |  |
| $\vdash$ | $\forall x (\mathcal{P}(x) \wedge \mathcal{Q}(x))$             |  |

**Example.**

$$\neg (\forall x \mathcal{P}(x)) \vdash \exists x (\neg \mathcal{P}(x))$$

|          |                                        |                                |
|----------|----------------------------------------|--------------------------------|
| 1        | $\neg (\forall x \mathcal{P}(x))$      |                                |
| 2        | $\neg (\exists x \neg \mathcal{P}(x))$ | Proof by contradiction         |
| 3        | $\neg \mathcal{P}(c)$                  | Assumption, with $c$ arbitrary |
| 4        | $\exists x \neg \mathcal{P}(x)$        | $\exists$ introduction         |
| 5        | $\perp$                                | Contradiction between 2 and 4  |
| 6        | $\mathcal{P}(c)$                       | From 2–5                       |
| 7        | $\forall x \mathcal{P}(x)$             | $\forall$ introduction         |
| 8        | $\perp$                                | Contradiction between 1 and 7  |
| $\vdash$ | $\exists x \neg \mathcal{P}(x)$        | By contradiction from 2–8      |

### 3. Details of the rules

We revisit the rules one by one with more explanations and examples.

#### 3.1. Rule I. $\forall$ elimination

$$\boxed{\forall x \mathcal{P}(x) \vdash \mathcal{P}(c)}$$

*Conditions.*

- If  $c$  is a constant in the language, there is no condition provided that the substitution of  $c$  for  $x$  is valid.
- If  $c$  is a term, none of its variables may become bound (captured) by a quantifier in  $\mathcal{P}$  during the substitution.

*Other formulations.*

$$\forall x \mathcal{P}(x) \vdash \mathcal{P}[c/x]$$

We also sometimes find the slightly abusive notation  $\forall x \mathcal{P}(x) \vdash \mathcal{P}(x)$ .

And here is the natural deduction schema:

$$\vdash \frac{\forall x \mathcal{P}(x)}{\mathcal{P}(c)}$$

*Examples.*

- $\forall x \mathcal{P}(x) \wedge Q(x, c) \vdash \mathcal{P}(c) \wedge Q(c, c)$  is a correct proof. We replaced the occurrences of  $x$  by a constant  $c$ , even though this constant already appeared in the formula.

- $\forall x P(x) \vdash P(f(a, b))$  is correct if  $f$  is a function and  $a, b$  are two constants.
- $\forall x P(x) \vdash P(y)$  is also correct, provided that  $y$  is not captured by a quantifier in  $\mathcal{P}$ .

*Counterexamples.*

- $(\forall x P(x)) \rightarrow Q(a, b)$  does not prove  $P(c) \rightarrow Q(a, b)$ , because the “for every” is not the main connective of the sentence. On the other hand,  $\forall x (P(x) \rightarrow Q(a, b))$  would indeed prove  $P(c) \rightarrow Q(a, b)$ .
- $\forall x \exists y Q(x, y)$  does not prove  $\exists y Q(y, y)$ . Indeed, in the formula “ $\exists y Q(x, y)$ ”, the variable  $x$  occurs as a free variable. Substituting  $y$  for  $x$  is therefore forbidden because the variable  $y$  inserted during the replacement would be captured by the formula’s quantifier  $\exists y$ .

*Remark.* How can we speak of a constant  $c$  if the language  $\mathcal{L}$  has no constant with that name, or has no constants at all? We allow ourselves to extend the language  $\mathcal{L}$  to a language  $\mathcal{L}' = \mathcal{L} \cup \{c\}$ , where  $c$  is a new constant symbol.

### 3.2. Rule II. $\forall$ introduction

$$\boxed{\mathcal{P}(c) \vdash \forall x \mathcal{P}(x)}$$

*Conditions.* The delicate point in using it is that  $c$  must be an arbitrary constant, that is, a constant whose symbol does not occur in any assumption currently in use.

*Other formulations.*

$$\begin{array}{c} \mathcal{P}(x) \vdash \forall x \mathcal{P}(x) \\ \vdash \begin{array}{c|c} \mathcal{P}(c) \\ \hline \forall x \mathcal{P}(x) \end{array} \end{array}$$

Let  $\Gamma$  be a set of formulas and  $c$  a constant that does not occur in the formulas of  $\Gamma$ .

$$\text{If } \Gamma \vdash \mathcal{P}(c) \quad \text{then} \quad \Gamma \vdash \forall x \mathcal{P}(x)$$

*Examples.*

- $c > 0 \vdash \forall x x > 0$ , provided that no assumption concerns  $c$ .
- $f(a, b) \neq 0 \vdash \forall x \forall y f(x, y) \neq 0$ , provided that no assumption concerns either  $a$  or  $b$ .

*Counterexamples.* Many things are forbidden, and it is easy to make a mistake when applying this rule.

- $c > 0, P(c)$  does not prove  $\forall x P(x)$  because the constraint  $c > 0$  means that  $c$  is not an arbitrary constant. On the other hand, if we had a proof of  $P(c)$  from the temporary assumption  $c > 0$  (with  $c$  otherwise arbitrary), we could prove  $\forall x (x > 0 \rightarrow P(x))$ .
- Here is almost the same example in natural deduction notation, along with another example:

|                                                                                                |                                                                                                                  |                                                                   |
|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| $\vdash \begin{array}{c c} c > 0 \\ \hline P(c) \\ \hline \cancel{\forall x P(x)} \end{array}$ | $\vdash \begin{array}{c c} \forall x Q(c, x) \\ \hline Q(c, c) \\ \hline \cancel{\forall x Q(x, x)} \end{array}$ | Valid $\forall$ elimination<br><br>Invalid $\forall$ introduction |
|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|

- Replacing only some occurrences of  $c$  is forbidden. For example,  $c = c$  does not prove  $\forall x x = c$ .

*Remark.* It is best if the symbol  $c$  used in the rule has not appeared at all earlier in the proof. But it is not always that simple. Here is a valid example:

|          |                                                                   |                                             |
|----------|-------------------------------------------------------------------|---------------------------------------------|
|          | $\vdots$                                                          | $c$ does not appear earlier                 |
|          | $\left  \begin{array}{c} P(c) \\ \hline Q(c) \end{array} \right.$ | Temporary assumption, condition on $c$      |
|          | $P(c) \rightarrow Q(c)$                                           | Conclusion in this case                     |
|          | $P(c) \rightarrow Q(c)$                                           | No assumption currently in use contains $c$ |
| $\vdash$ | $\forall x (P(x) \rightarrow Q(x))$                               | $\forall$ introduction                      |

### 3.3. Rule III. $\exists$ introduction

$$\boxed{P(c) \vdash \exists x P(x)}$$

*Conditions.*  $c$  is a constant or a term with no free variable.

*Other formulations.*

$$\begin{array}{c} P[c/x] \vdash \exists x P(x) \\ \vdash \left| \begin{array}{c} P(c) \\ \hline \exists x P(x) \end{array} \right. \end{array}$$

*Examples.*

- $c^2 \neq c \vdash \exists x x^2 \neq x$  (seen above).
- $c^2 \neq c \vdash \exists x x^2 \neq c$  (partial replacement allowed)
- $c^2 \neq c \vdash \exists x c^2 \neq x$  (partial replacement)
- $a \times b = 12 \vdash \exists x \exists y x \times y = 12$  (two applications of the rule)

*Remark.* There is no danger with this rule. However, care must be taken to introduce the “there exists” at the beginning of the resulting formula. For example, the following is correct:

$$\forall x Q(x, b) \vdash \exists y \forall x Q(x, y)$$

But the same premise would not prove  $\forall x \exists y Q(x, y)$ .

*Nonempty domain.* In first-order logic, we will always assume that we are working with a nonempty domain.

### 3.4. Rule IV. $\exists$ elimination

$$\boxed{\exists x P(x), [P(c) \vdash Q] \vdash Q}$$

The notation in brackets corresponds to a subproof.

*Conditions.*  $c$  must be a **fresh constant**: first, it must not occur in any assumption currently in use; second, it must not occur in the formula  $Q$  (nor, of course, in  $P(x)$ ). It is best if the constant occurs neither earlier in the proof nor in  $Q$ .

*Other formulations.*

$$\begin{aligned} \exists x \, P(x), [P[c/x] \vdash Q] &\vdash Q \\ \exists x \, P(x), [P(x) \vdash Q] &\vdash Q \end{aligned}$$

Let  $\Gamma$  be a set of formulas and  $c$  a constant of the language that occurs neither in  $Q$  nor in the formulas of  $\Gamma$ :

$$\text{If } \Gamma \vdash \exists x \, P(x) \text{ and } P(c) \vdash Q, \text{ then } \Gamma \vdash Q$$

The natural deduction notation is the clearest.

$$\begin{array}{c|c|c} & \exists x \, P(x) & \\ \hline & P(c) & \text{Temporary assumption} \\ & \vdots & \\ & Q & \\ \hline \vdash & Q & \end{array}$$

*Example.* Let us see how to use this rule in practice:

$$\exists x \, f(x) = 0, \forall x \, (f(x) = 0 \rightarrow g(x) = 1) \vdash \exists x \, g(x) = 1$$

This is not a direct application of the rule, but it is close:

$$\begin{array}{c|c|c} \exists x \, f(x) = 0 & & \\ \forall x \, (f(x) = 0 \rightarrow g(x) = 1) & & \\ \hline f(c) = 0 & \text{Temporary assumption} & \\ \hline f(c) = 0 \rightarrow g(c) = 1 & \forall \text{ elimination} & \\ \hline g(c) = 1 & \text{By implication} & \\ \hline \exists x \, g(x) = 1 & \text{This is the formula } Q \text{ in the rule} & \\ \hline \vdash \exists x \, g(x) = 1 & \exists \text{ elimination} & \end{array}$$

*Remark.* If we have several successive “there exists” statements, we must take a new fresh constant for each one. For example, if we have:

$$\exists x \, P(x), \exists y \, Q(y)$$

then in a proof we cannot choose  $P(c)$  and  $Q(c)$  with the same  $c$ . We must use two fresh constants  $c$  and  $d$  in order to write  $P(c)$  and  $Q(d)$  in the temporary assumptions.

### 3.5. Rule V. Equality

$$\boxed{t = s, P(t) \vdash P(s)}$$

*Conditions.*  $t$  and  $s$  are terms in the language. The substitution of  $t$  and  $s$  for  $x$  must be valid. There are no other conditions.

Other formulations.

$$x = y, \mathcal{P}(x) \vdash \mathcal{P}(y)$$

$$\vdash \frac{\begin{array}{c} t = s \\ \mathcal{P}(t) \end{array}}{\mathcal{P}(s)}$$

Example.  $t = x + 1, t^2 \neq 0 \vdash (x + 1)^2 \neq 0$ .

Remark. This rule can be replaced by the following axioms for the equality symbol:

- $\forall x \ x = x$
- $\forall x \ \forall y \ ((x = y) \rightarrow (\mathcal{P}(x) \leftrightarrow \mathcal{P}(y)))$

## Exercises

### Proofs in first-order logic

**Exercise 1.** Write proofs of the following theorems.  $P, Q, R$  denote predicates, and  $f, g$  denote functions.

1.  $\forall x \ P(x), \ P(a) \rightarrow Q(a) \vdash Q(a)$
2.  $\forall x \ P(x) \wedge Q(x) \vdash \exists x \ P(x)$
3.  $\forall x \ \forall y \ R(x, y) \vdash \forall x \ R(x, x)$
4.  $\exists x \ f(x) = g(x) \vdash \exists x \ \exists y \ f(x) = g(y)$

**Exercise 2.** Write proofs of the following theorems.  $P, Q, R$  denote predicates, and  $f$  denotes a function.

1.  $\vdash \forall x \ \forall y \ (f(x) \neq f(y) \rightarrow x \neq y)$
2.  $\forall x \ P(x), \ \forall y \ Q(y) \vdash \exists z \ P(z) \wedge Q(z)$
3.  $\forall x \ P(x) \rightarrow Q(x), \ \forall y \ Q(y) \rightarrow R(y) \vdash \forall z \ P(z) \rightarrow R(z)$
4.  $\forall x \ (P(x) \rightarrow Q), \ \exists x \ P(x) \vdash Q$ , where  $Q$  does not depend on  $x$ .

### Details of the rules

**Exercise 3.** Explain the error made in applying the  $\forall$  elimination rule.

1.  $\exists x \ \forall y \ Q(x, y)$  directly proves  $\exists x \ Q(x, x)$ . Bonus: give a correct proof.
2.  $\forall x \ (P(x) \rightarrow Q(c))$  proves  $P(c)$ .
3.  $\forall x \ \exists y \ Q(x, y)$  proves  $\exists y \ Q(y, y)$ .

**Exercise 4.** Explain the error made in applying the  $\forall$  introduction rule.

1.  $c = 1, \ c > 0$  proves  $\forall x \ x > 0$ .
2.  $\exists x \ P(x)$  proves  $\forall x \ P(x)$ .
3.  $(x \neq 0) \rightarrow (x^2 \neq 0)$  proves  $\forall x \ x^2 \neq 0$ .
4.  $(x = 0) \vee (x \neq 0)$  proves  $(\forall x \ x = 0) \vee (\forall x \ x \neq 0)$ .
5.  $f(c) = g(c)$  proves  $\forall x \ f(x) = g(c)$ .

**Exercise 5.** Explain the error made in applying the  $\exists$  introduction rule.

1.  $\forall x P(x)$  directly proves  $\exists x P(x)$ .
2.  $P(a) \wedge Q(a)$  proves  $\exists x P(x) \wedge Q(y)$ .
3.  $c = 0, \exists x P(x)$  proves  $\exists x (x = 0) \wedge P(x)$ .
4.  $\forall x Q(y, y)$  proves  $\exists x \forall y Q(x, y)$ .
5.  $\forall x Q(y, y)$  proves  $\forall y \exists x Q(x, y)$ .

**Exercise 6.** Explain the error made in applying the  $\exists$  elimination rule.

1.  $\exists x P(x), P(0) \rightarrow Q$  proves  $Q$ .
2.  $\exists x P(x)$  proves  $P(c)$ .
3.  $\exists x (P(x) \rightarrow Q), P(c)$  proves  $Q$ .
4.  $\exists x P(x), \exists y Q(y)$  proves  $\exists z P(z) \wedge Q(z)$ .

### Negation of quantifiers and derived rules

**Exercise 7.** Write what the following formulas prove. The formula proved must no longer contain any negation to the left of a quantifier.

1.  $\neg(\forall x f(x) = g(x))$
2.  $\neg(\exists x P(x) \leftrightarrow Q(x))$
3.  $\neg[\forall x (P(x) \rightarrow \exists y Q(y))]$

**Exercise 8.** Prove all the derived rules using the basic rules.

# Soundness Theorem

We connect “proof” and “truth” through the Soundness Theorem, which states that every provable formula is true.

## 1. The Soundness Theorem

### 1.1. Statement

Let  $\mathcal{L}$  be a fixed first-order language.

**Theorem 1** (Soundness Theorem).

$$\boxed{\text{If } \mathcal{P}_1, \dots, \mathcal{P}_l \vdash Q \text{ then } \mathcal{P}_1, \dots, \mathcal{P}_l \models Q.}$$

More generally, if  $\Gamma$  is a (possibly infinite) set of formulas:

$$\text{If } \Gamma \vdash Q \text{ then } \Gamma \models Q.$$

This is a common-sense theorem: a proven formula is a true formula! This result is implicit in Gödel’s 1930 thesis, in which he focuses on the converse (the Completeness Theorem). The idea is that the inference rules for proofs were well chosen, since each rule preserves the true/false nature of first-order logic.

### 1.2. Review

Recall that

$$\mathcal{P}_1, \dots, \mathcal{P}_l \vdash Q$$

means that  $Q$  can be deduced from the formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  in finitely many steps, each of which consists in applying an inference rule (from a finite, precisely specified collection).

Also recall that

$$\mathcal{P}_1, \dots, \mathcal{P}_l \models Q$$

means that, for every structure  $S$  and every assignment  $s$ , if  $\mathcal{P}_1, \dots, \mathcal{P}_l$  are true under this interpretation, then  $Q$  is also true under this interpretation.

Thus, the Soundness Theorem states that a provable formula is true under every interpretation, independently of the chosen structure.

It can also be restated as follows: if  $\mathcal{P} \vdash \perp$ , then  $\mathcal{P}$  is unsatisfiable (that is,  $\mathcal{P}$  is a contradiction); in other words, regardless of  $S$  and  $s$ , the interpretation of  $\mathcal{P}$  is false. Recall that  $\perp$  denotes the formula that is always false. Here is a proof of this restatement: suppose there are  $S$  and  $s$  for which  $\mathcal{P}$  is true. Since  $\perp$  is false under every interpretation, this contradicts  $\mathcal{P} \models_{S,s} \perp$ . Thus,  $\mathcal{P}$  is not satisfiable.

### 1.3. Applications

Let us explain the usefulness of the Soundness Theorem.

1. In one sense,  $\mathcal{P} \vdash \mathcal{Q}$  is easy. Indeed, the task is to construct a proof. Although finding a proof is by no means always easy, it is still a process with finitely many steps and finitely many possible rules.
2. By contrast, showing that no proof establishes that  $\mathcal{P}$  implies  $\mathcal{Q}$ , that is, that  $\mathcal{P} \not\vdash \mathcal{Q}$ , is difficult. Failing to find a proof does not mean that none exists.
3. On the other hand, showing that  $\mathcal{P} \models \mathcal{Q}$  is not straightforward, because this would require checking infinitely many conceivable structures and assignments.
4. Conversely, establishing  $\mathcal{P} \not\models \mathcal{Q}$  is easy. It is enough to provide a counterexample by specifying a structure and an assignment such that  $\mathcal{P}$  is true and  $\mathcal{Q}$  is false under this interpretation.

In summary, the Soundness Theorem can help with the difficult cases, either directly or through its contrapositive:

$$\begin{array}{ll} \text{If } \mathcal{P} \vdash \mathcal{Q} \text{ then } \mathcal{P} \models \mathcal{Q}. \\ \text{If } \mathcal{P} \not\vdash \mathcal{Q} \text{ then } \mathcal{P} \not\models \mathcal{Q}. \end{array}$$

## 2. Proof of the Soundness Theorem

We are going to prove that each inference rule preserves true/false logic and the satisfaction of formulas. Since a proof is a finite sequence of applications of these rules, the Soundness Theorem will then follow by induction.

### 2.1. Rules of true/false logic

See the chapter “Proofs” in Part One for the complete list of these rules. Here we give only a few examples.

- We have the rule “ $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$ ”, and indeed the logical consequence “ $\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$ ” holds.
- We have the rule “ $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$ ”, and indeed the logical consequence “ $\mathcal{P}, \mathcal{Q} \models \mathcal{P} \wedge \mathcal{Q}$ ” holds.
- We have the rule “ $\neg\neg\mathcal{P} \vdash \mathcal{P}$ ”, and indeed the logical consequence “ $\neg\neg\mathcal{P} \models \mathcal{P}$ ” holds.
- We have the rule “ $\perp \vdash \mathcal{P}$ ”, and indeed the logical consequence “ $\perp \models \mathcal{P}$ ” holds.
- We have the rule “ $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}$ ”, and indeed the logical consequence “ $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \models \mathcal{Q}$ ” holds.

These logical consequences are easily justified using truth tables. For example, to prove  $\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$ , fix a structure  $\mathcal{S}$  and an assignment  $s$ , and draw up the truth table:

| $\mathcal{P}$ | $\mathcal{Q}$ | $\mathcal{P} \wedge \mathcal{Q}$ | $\mathcal{P}$ |
|---------------|---------------|----------------------------------|---------------|
| T             | T             | T                                | T             |
| T             | F             | F                                | —             |
| F             | T             | F                                | —             |
| F             | F             | F                                | —             |

Whenever  $\mathcal{P} \wedge \mathcal{Q}$  is true,  $\mathcal{P}$  is also true. Thus,

$$\mathcal{P} \wedge \mathcal{Q} \models_{\mathcal{S},s} \mathcal{P}$$

and since this is true for all  $\mathcal{S}$  and  $s$ :

$$\mathcal{P} \wedge \mathcal{Q} \models \mathcal{P}$$

The same can be done for *modus ponens*:

| $\mathcal{P}$ | $\mathcal{Q}$ | $\mathcal{P} \rightarrow \mathcal{Q}$ | $\mathcal{Q}$ |
|---------------|---------------|---------------------------------------|---------------|
| T             | T             | T                                     | T             |
| T             | F             | F                                     | —             |
| F             | T             | T                                     | —             |
| F             | F             | T                                     | —             |

Whenever  $\mathcal{P}$  is true and  $\mathcal{P} \rightarrow \mathcal{Q}$  is also true,  $\mathcal{Q}$  is true as well.

## 2.2. Rules for quantifiers

The rules involving quantifiers are also compatible with satisfaction. This again follows from the fact that the inference rules were chosen sensibly. However, establishing their validity involves manipulations that are not always easy to set out in detail. On a first reading, you may skip this step and go directly to the end of the proof. Studying these proofs, however, helps clarify the conditions under which the inference rules may be applied.

Recall the definitions and notions from the chapter “Satisfaction”.

$$\models_{\mathcal{S},s} \forall x \mathcal{P}(x) \quad \text{iff} \quad \text{for every } d \in E, \models_{\mathcal{S},s[x \mapsto d]} \mathcal{P}(x)$$

The notation  $s[x \mapsto d]$  denotes the assignment that maps  $x$  to  $d$  and agrees with  $s$  on all other variables.

$$\models_{\mathcal{S},s} \exists x \mathcal{P}(x) \quad \text{iff} \quad \text{there exists } d \in E, \models_{\mathcal{S},s[x \mapsto d]} \mathcal{P}(x)$$

Recall the relationship between assignments and substitution:

$$\models_{\mathcal{S},s} \mathcal{P}[c/x] \quad \text{iff} \quad \models_{\mathcal{S},s[x \mapsto c^{\mathcal{S}}]} \mathcal{P}(x)$$

Finally, for a formula  $\mathcal{P}$ , we write  $\mathcal{P}(x)$  to emphasize its dependence on the variable  $x$ , and we write:

$$\mathcal{P}(c) = \mathcal{P}[c/x]$$

Writing  $\mathcal{P}(c)$  presupposes that substituting  $c$  for  $x$  is valid.

We begin with the two simple rules.

**Rule I.** We have the rule

$$\forall x \mathcal{P}(x) \vdash \mathcal{P}(c)$$

where  $c$  is a constant. And indeed, the corresponding logical consequence holds:

$$\forall x \mathcal{P}(x) \models \mathcal{P}(c)$$

*Proof.* Consider a structure  $\mathcal{S}$  with domain  $E$  and an assignment  $s$  such that “ $\forall x \mathcal{P}(x)$ ” is true. By definition, this means that  $\mathcal{P}(d)$  is true for every  $d$  in the domain  $E$ . In particular,  $\mathcal{P}(c)$  is therefore satisfied.

Let us now give the formal proof. Let  $c$  be a constant of the language, and let  $c^{\mathcal{S}}$  be its interpretation in  $E$ .

$$\begin{array}{ll} & \models_{\mathcal{S},s} \forall x \mathcal{P}(x) \\ \text{therefore} & \models_{\mathcal{S},s[x \mapsto d]} \mathcal{P}(x) \quad \text{for every } d \in E \\ \text{in particular} & \models_{\mathcal{S},s[x \mapsto c^{\mathcal{S}}]} \mathcal{P}(x) \\ \text{therefore} & \models_{\mathcal{S},s} \mathcal{P}[c/x] \\ \text{hence} & \models_{\mathcal{S},s} \mathcal{P}(c) \end{array}$$

□

**Rule III.** We have the rule

$$\mathcal{P}(c) \vdash \exists x \mathcal{P}(x)$$

where  $c$  is a constant, and indeed the corresponding logical consequence holds:

$$\mathcal{P}(c) \models \exists x \mathcal{P}(x)$$

*Proof.* Fix  $S$  with domain  $E$ , and fix  $s$ . If  $\mathcal{P}(c)$  is true, then there is a  $d \in E$  (namely,  $d = c^S$ ) such that  $\mathcal{P}(d)$  is true; therefore, “ $\exists x \mathcal{P}(x)$ ” is true.

Let us now give the formal proof. Let  $c$  be a constant of the language, and let  $c^S$  be its interpretation in  $E$ .

$$\begin{array}{ll} & \models_{S,s} \mathcal{P}(c) \\ \text{therefore} & \models_{S,s} \mathcal{P}[c/x] \\ \text{therefore} & \models_{S,s[x \mapsto c^S]} \mathcal{P}(x) \\ \text{hence} & \models_{S,s} \exists x \mathcal{P}(x) \quad \text{since the formula is true for } d = c^S \end{array}$$

□

**Rule II.** We restate this rule as follows. Let  $\Gamma$  be a set of formulas, and let  $c$  be a constant that does not occur in the formulas of  $\Gamma$ . Rule II is written:

$$\text{If } \Gamma \vdash \mathcal{P}(c) \quad \text{then} \quad \Gamma \vdash \forall x \mathcal{P}(x)$$

And indeed there is a corresponding statement in terms of satisfaction:

$$\text{If } \Gamma \models \mathcal{P}(c) \quad \text{then} \quad \Gamma \models \forall x \mathcal{P}(x)$$

*Proof.* We have a language  $\mathcal{L}$  with a constant  $c$ . Suppose  $\Gamma \models \mathcal{P}(c)$ . Fix a structure  $S$  with domain  $E$  and an assignment  $s$  such that  $\Gamma$  is satisfied. Intuitively, we must show that  $\mathcal{P}(d)$  is satisfied for every  $d \in E$ . Formally, we must prove that  $\models_{S,s[x \mapsto d]} \mathcal{P}(x)$  for every  $d \in E$ .

So far, we have a structure  $S$  with domain  $E$ ; in this structure, the constant  $c$  of the language is interpreted as a value  $c^S \in E$ . The assumptions then imply that  $\mathcal{P}(c^S)$  is satisfied. We have therefore verified  $\mathcal{P}(x)$  for one value in the domain, but we want to prove it for every value.

Fix an arbitrary value  $d \in E$  in the domain. We consider another structure  $S'$  with the same domain  $E$ . The structure  $S'$  is the same as  $S$ , except that the interpretation of  $c$  in this structure is our fixed value  $d$ :  $c^{S'} = d$ . We do not change the assignment  $s$ . Since the formulas of  $\Gamma$  do not depend on the constant  $c$ , and since  $\Gamma$  is satisfied under  $S$  and  $s$ , it is also satisfied under  $S'$  and  $s$ .

The assumption  $\Gamma \models \mathcal{P}(c)$  expresses satisfaction for every structure, so in particular we have

$$\models_{S',s} \mathcal{P}(c)$$

Since  $\Gamma$  is satisfied under  $S'$  and  $s$ , we have:

$$\begin{array}{ll} & \models_{S',s} \mathcal{P}(c) \\ \text{therefore} & \models_{S',s} \mathcal{P}[c/x] \\ \text{therefore} & \models_{S',s[x \mapsto c^{S'}]} \mathcal{P}(x) \\ \text{therefore} & \models_{S',s[x \mapsto d]} \mathcal{P}(x) \\ \text{hence} & \models_{S,s[x \mapsto d]} \mathcal{P}(x) \end{array}$$

Note that in the last line we returned to the structure  $S$ . This is possible because the constant  $c$  does not occur in the formula  $\mathcal{P}(x)$  (indeed, writing  $\mathcal{P}(c)$  requires the substitution of  $c$  for  $x$  to be possible, and hence  $c$  not to occur in  $\mathcal{P}(x)$ ).

Thus, for every  $d \in E$ , we have obtained:

$$\models_{S,s[x \mapsto d]} \mathcal{P}(x)$$

This is precisely the definition of:

$$\models_{S,s} \forall x \mathcal{P}(x)$$

□

**Rule IV.** We restate this rule as follows. Let  $\mathcal{Q}$  denote a formula, let  $\Gamma$  be a set of formulas, and let  $c$  be a constant of the language that occurs neither in  $\mathcal{Q}$  nor in the formulas of  $\Gamma$ . Rule IV is written:

$$\text{If } \Gamma \vdash \exists x \mathcal{P}(x) \text{ and } \mathcal{P}(c) \vdash \mathcal{Q} \text{ then } \Gamma \vdash \mathcal{Q}$$

And indeed there is a corresponding statement in terms of satisfaction:

$$\text{If } \Gamma \models \exists x \mathcal{P}(x) \text{ and } \mathcal{P}(c) \models \mathcal{Q} \text{ then } \Gamma \models \mathcal{Q}$$

*Proof.* First consider a structure  $S$  with domain  $E$  and an assignment  $s$  such that  $\Gamma$  is satisfied. Then, by assumption, “ $\exists x \mathcal{P}(x)$ ” is satisfied, so there exists  $d \in E$  such that  $\mathcal{P}(x)$  is satisfied under the assignment  $s[x \mapsto d]$ .

Change the structure  $S$  into a structure  $S'$  in which only the interpretation of the constant  $c$  is changed: in  $S'$ , define  $c^{S'} = d$ . Since  $\mathcal{Q}$  and the formulas of  $\Gamma$  do not depend on  $c$ , their satisfaction in  $S$  and in  $S'$  is equivalent. Thus,  $\Gamma$  is satisfied in  $S'$ , and  $\mathcal{P}(c)$  is satisfied as well:

$$\begin{aligned} & \models_{S',s} \exists x \mathcal{P}(x) \\ \text{therefore } & \models_{S',s[x \mapsto d]} \mathcal{P}(x) \quad \text{for the same } d \in E \text{ as above} \\ \text{therefore } & \models_{S',s[x \mapsto c^{S'}]} \mathcal{P}(x) \quad \text{since } c^{S'} = d \\ \text{therefore } & \models_{S',s} \mathcal{P}[c/x] \\ \text{therefore } & \models_{S',s} \mathcal{P}(c) \\ \text{therefore } & \models_{S',s} \mathcal{Q} \quad \text{by the assumption } \mathcal{P}(c) \models \mathcal{Q} \\ \text{hence } & \models_{S,s} \mathcal{Q} \quad \text{since } c \text{ does not occur in } \mathcal{Q} \end{aligned}$$

□

### 2.3. Proof of the theorem

Let  $\Gamma = \{\mathcal{P}_1, \dots, \mathcal{P}_l\}$  denote the set of formulas serving as assumptions, and let  $\mathcal{Q}$  be the conclusion formula. Suppose  $\Gamma \vdash \mathcal{Q}$ . We must prove that  $\Gamma \models \mathcal{Q}$ .

The statement  $\Gamma \vdash \mathcal{Q}$  means that there is a proof of  $\mathcal{Q}$  from the formulas in  $\Gamma$ . Suppose this proof is written in natural deduction as in the figure below. The formulas  $\mathcal{P}_1, \dots, \mathcal{P}_l$  are the assumptions. Each formula  $\mathcal{P}_k$ , with  $l+1 \leq k \leq q$ , is obtained by applying one of the inference rules to the preceding lines  $\mathcal{P}_1, \dots, \mathcal{P}_{k-1}$ . The final formula obtained,  $\mathcal{P}_q$ , is the formula  $\mathcal{Q}$ .

|          |                               |             |
|----------|-------------------------------|-------------|
|          | $\mathcal{P}_1$               |             |
|          | $\vdots$                      | Assumptions |
|          | $\mathcal{P}_l$               |             |
|          | <hr/>                         |             |
|          | $\mathcal{P}_{l+1}$           |             |
|          | $\vdots$                      |             |
|          | $\mathcal{P}_k$               | Proof steps |
|          | $\vdots$                      |             |
|          | $\mathcal{P}_{q-1}$           |             |
| $\vdash$ | $\mathcal{P}_q = \mathcal{Q}$ | Conclusion  |

Fix a structure  $\mathcal{S}$  and an assignment  $s$  such that  $\Gamma$  is satisfied. We must show that  $\mathcal{Q}$  is then also satisfied (under  $\mathcal{S}$  and  $s$ ). We prove this by induction on  $k$  (for  $1 \leq k \leq q$ ):

$\mathcal{H}_k$  :      the formula  $\mathcal{P}_k$  is satisfied

- *Base case.* For  $k = 1, \dots, l$ , the formula  $\mathcal{P}_k$  is one of the formulas in  $\Gamma$ , so it is satisfied by assumption.
- *Induction step.* Let  $k$  satisfy  $1 \leq k < q$ . Suppose  $\mathcal{P}_1, \dots, \mathcal{P}_k$  are satisfied (under  $\mathcal{S}$  and  $s$ ), and show that  $\mathcal{P}_{k+1}$  is also satisfied. Indeed,  $\mathcal{P}_{k+1}$  is obtained from the preceding formulas by applying one of the inference rules for proofs. We have already shown that each of these rules preserves the satisfaction of formulas, so  $\mathcal{P}_{k+1}$  is satisfied (under  $\mathcal{S}$  and  $s$ ).
- *Conclusion.* By the principle of induction, the final formula  $\mathcal{P}_q$ , which is  $\mathcal{Q}$ , is satisfied.

Thus,  $\models_{\mathcal{S}, s} \mathcal{Q}$ , and since  $\mathcal{S}$  and  $s$  are arbitrary,  $\Gamma \models \mathcal{Q}$ .

## Exercises

### The Soundness Theorem

**Exercise 1.** Let  $\Gamma$  be a set of formulas such that  $\Gamma \vdash \perp$ . Show that  $\Gamma$  is a contradiction; that is, there is no structure  $\mathcal{S}$  and no assignment  $s$  under which all the formulas in  $\Gamma$  are satisfied.

**Exercise 2.** Prove the unprovability of the following statements:

1.  $\exists x P(x) \not\vdash \forall x P(x)$
2.  $\forall x \exists y Q(x, y) \not\vdash \exists y \forall x Q(x, y)$
3.  $\exists x (P(x) \rightarrow Q(x)) \not\vdash \exists x (Q(x) \rightarrow P(x))$
4.  $\forall x (P(x) \vee Q(x)) \not\vdash (\forall x P(x)) \vee (\forall x Q(x))$

**Exercise 3.** Prove the following logical consequences:

1.  $\forall x (P(x) \rightarrow Q(x)), P(c) \models Q(c)$
2.  $\forall x (P(x) \rightarrow \perp) \models \neg(\exists x P(x))$
3.  $\forall x (P(x) \leftrightarrow Q(x)), \forall x (Q(x) \leftrightarrow R(x)) \models \forall x (P(x) \leftrightarrow R(x))$
4.  $\forall x (P(x) \wedge Q(x)) \models \forall x P(x)$

## Proofs of the Soundness Theorem

**Exercise 4.** Verify that all the basic rules of true/false logic are compatible with satisfaction.

# Completeness Theorem

*The Completeness Theorem states that, in first-order logic, everything that is true is provable. We state this theorem, its variants, and its applications.*

## 1. Completeness Theorem

### 1.1. Statement

Let  $\mathcal{L}$  be a first-order language.

**Theorem 1** (Completeness Theorem).

$$\boxed{\text{If } \mathcal{P}_1, \dots, \mathcal{P}_l \models \mathcal{Q} \text{ then } \mathcal{P}_1, \dots, \mathcal{P}_l \vdash \mathcal{Q}}$$

This can be read as follows: if  $\mathcal{Q}$  is satisfied whenever the assumptions  $\mathcal{P}_1, \dots, \mathcal{P}_l$  are satisfied, then there is a proof of  $\mathcal{Q}$  from  $\mathcal{P}_1, \dots, \mathcal{P}_l$ .

Here is the variant for a set  $\Gamma$  of formulas, which may be countably infinite:

**Theorem 2** (Completeness Theorem).

$$\text{If } \Gamma \models \mathcal{Q} \text{ then } \Gamma \vdash \mathcal{Q}.$$

The conclusion  $\Gamma \vdash \mathcal{Q}$  is an existence result: “there is a proof of  $\mathcal{Q}$  from the formulas in  $\Gamma$ ”. However, the theorem is not constructive and gives no concrete information about this proof.

The proof of the Completeness Theorem is fairly demanding. It has points in common with the Compactness Theorem for true/false logic, but with an additional level of difficulty. The proof will be given in the next chapter. We will now focus on the different restatements and applications.

### 1.2. Equivalence

In the previous chapter, we saw the Soundness Theorem, which is exactly the converse of the Completeness Theorem. Thus, we obtain an equivalence that is also called the Completeness Theorem:

**Theorem 3** (Completeness Theorem).

$$\boxed{\Gamma \models \mathcal{Q} \text{ iff } \Gamma \vdash \mathcal{Q}}$$

Thus, in first-order logic, logical consequence and provability are equivalent notions.

### 1.3. What about incompleteness?

Is there not a contradiction between obtaining the Completeness Theorem and, at the end of the book, obtaining the Incompleteness Theorem, whose name seems to suggest the opposite? In fact, the Completeness Theorem applies to first-order logic, which is a fairly rudimentary logic, but it uses the quite strong assumption  $\Gamma \models \mathcal{Q}$ , since logical consequence requires the formula to be true in every possible structure.

The Incompleteness Theorem does not concern all structures, but rather more specific theories, typically ones that contain the natural numbers  $\mathbb{N}$  together with the axioms of arithmetic (the Peano axioms). We will see that, for the arithmetic of the natural numbers, there are formulas  $\mathcal{Q}$  such that neither  $\mathcal{Q}$  nor  $\neg\mathcal{Q}$  is provable. In other words, there are true but unprovable formulas, but all this holds only for the structure associated with the arithmetic of the natural numbers. This does not contradict the Completeness Theorem, because the assumption “ $\Gamma \models \mathcal{Q}$ ” is not verified for every structure, but only for certain specific structures; the conditions of the Completeness Theorem are therefore not met.

Thus, the Completeness and Incompleteness Theorems are both true, but they do not apply to the same situations and therefore do not contradict each other.

## 2. Applications to theories

We will recall some definitions and introduce new notions. Let  $\mathcal{L}$  be a first-order language.

### 2.1. Consistent theory, model

- A **closed formula** (also called a **sentence**) is a formula with no free variables; that is, every occurrence of a variable lies within the scope of a quantifier.
- A **theory**  $\mathcal{T}$  is a set of closed formulas. (Note: some authors call any set of formulas a “theory”.)
- A structure  $\mathcal{S}$  is a **model** of  $\mathcal{T}$  if  $\mathcal{T}$  is satisfied in  $\mathcal{S}$ , denoted  $\models_{\mathcal{S}} \mathcal{T}$ ; that is, the interpretation of every formula in  $\mathcal{T}$  is true in  $\mathcal{S}$ . (Since the formulas are closed, there is no assignment to consider.)

Here is a very important new notion.

#### Definition.

A theory  $\mathcal{T}$  is **inconsistent** if there is a formula  $\mathcal{Q}$  such that:

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{and} \quad \mathcal{T} \vdash \neg\mathcal{Q}.$$

Otherwise,  $\mathcal{T}$  is said to be **consistent**.

Equivalently:

|                                                                     |
|---------------------------------------------------------------------|
| $\mathcal{T} \text{ is inconsistent iff } \mathcal{T} \vdash \perp$ |
|---------------------------------------------------------------------|

where  $\perp$  denotes a contradiction. Thus, a contradiction can never be proved from a consistent theory (which is reassuring).

### 2.2. Natural deduction lemma

Let  $\Gamma$  denote a set of formulas, and let  $\mathcal{P}$  and  $\mathcal{Q}$  be formulas.

**Lemma 1** (Natural deduction lemma).

|                                                                                                                     |
|---------------------------------------------------------------------------------------------------------------------|
| $\Gamma, \mathcal{P} \vdash \mathcal{Q} \quad \text{iff} \quad \Gamma \vdash (\mathcal{P} \rightarrow \mathcal{Q})$ |
|---------------------------------------------------------------------------------------------------------------------|

*Proof.* The reverse implication  $\Leftarrow$  is simply the *modus ponens* rule  $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}$ .

The forward implication  $\Rightarrow$  is immediate when we have a proof in natural deduction (hence the name of the lemma); it is simply a matter of rewriting the proof:

**Proof of**  $\Gamma, \mathcal{P} \vdash \mathcal{Q}$

|          |               |             |
|----------|---------------|-------------|
| $\vdash$ | $\Gamma$      |             |
|          | $\mathcal{P}$ | Assumptions |
|          | $\vdots$      | Steps       |
|          | $\mathcal{Q}$ | Conclusion  |

**Proof of**  $\Gamma \vdash (\mathcal{P} \rightarrow \mathcal{Q})$

|          |                                       |                         |
|----------|---------------------------------------|-------------------------|
| $\vdash$ | $\Gamma$                              |                         |
|          | $\mathcal{P}$                         | Temporary assumption    |
|          | $\vdots$                              | Same steps              |
|          | $\mathcal{Q}$                         | Intermediate conclusion |
|          | $\mathcal{P} \rightarrow \mathcal{Q}$ | Conclusion              |

□

Let  $\mathcal{T}$  be a theory and  $\mathcal{Q}$  a closed formula.

**Corollary 1.**

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{iff} \quad \mathcal{T} \cup \{\neg \mathcal{Q}\} \text{ is inconsistent}$$

*Proof.*

$$\begin{aligned} & \mathcal{T} \cup \{\neg \mathcal{Q}\} \text{ is inconsistent} \\ \text{iff } & \mathcal{T} \cup \{\neg \mathcal{Q}\} \vdash \perp \quad \text{by definition} \\ \text{iff } & \mathcal{T} \vdash (\neg \mathcal{Q} \rightarrow \perp) \quad \text{by the natural deduction lemma} \\ \text{iff } & \mathcal{T} \vdash \neg(\neg \mathcal{Q}) \quad \text{since, by definition, } \neg \mathcal{P} \text{ means } \mathcal{P} \rightarrow \perp \\ \text{iff } & \mathcal{T} \vdash \mathcal{Q} \quad \text{by double negation} \end{aligned}$$

□

## 2.3. Different formulations of the Completeness Theorem

We will state the Completeness Theorem along with several restatements.

Setting:  $\mathcal{T}$  is a theory and  $\mathcal{Q}$  is a closed formula.

Standard formulation:

**Theorem 4** (Theorem A).

$$\mathcal{T} \models \mathcal{Q} \quad \text{iff} \quad \mathcal{T} \vdash \mathcal{Q}$$

Restatement:

**Theorem 5** (Theorem B).

$$\mathcal{T} \text{ is consistent} \quad \text{iff} \quad \mathcal{T} \text{ has a model}$$

We could state a similar result for any set of formulas:

$$\Gamma \text{ is consistent} \quad \text{iff} \quad \Gamma \text{ is satisfiable.}$$

## 2.4. Proof of Theorem B from Theorem A

$$\begin{aligned}
 & \mathcal{T} \text{ is inconsistent} \\
 \text{iff } & \mathcal{T} \vdash \perp \quad \text{by definition} \\
 \text{iff } & \mathcal{T} \models \perp \quad \text{by Theorem A} \\
 \text{iff } & \mathcal{T} \text{ is satisfied by no structure } \mathcal{S} \\
 \text{iff } & \mathcal{T} \text{ has no model}
 \end{aligned}$$

In an exercise (and in the next chapter), we will see that Theorem A can be proved from Theorem B.

## 3. Application to finiteness

We are interested in the case where the set of assumptions is infinite (but countable). We will see that only finitely many assumptions are needed. We will obtain two results: one about proofs (the finiteness theorem) and one about satisfaction (the Compactness Theorem for first-order logic, analogous to the theorem in the chapter “Compactness Theorem” for true/false logic).

### 3.1. Finiteness theorem

Let  $\Gamma$  be a (possibly infinite) set of formulas, and let  $\mathcal{Q}$  be a formula in a language  $\mathcal{L}$ .

**Theorem 6** (Finiteness theorem).

*If  $\Gamma \vdash \mathcal{Q}$ , then there is a finite subset  $\Gamma_0 \subset \Gamma$  such that  $\Gamma_0 \vdash \mathcal{Q}$ .*

Note: the converse is of course true.

*Proof.* The proof is based on the very nature of a proof. The statement  $\Gamma \vdash \mathcal{Q}$  means that there is a proof of  $\mathcal{Q}$  from the formulas in  $\Gamma$ . Consider such a proof. This proof consists of finitely many steps, and at each step a rule is applied. Each rule involves only finitely many formulas from  $\Gamma$ . Let  $\Gamma_0$  denote the set of formulas from  $\Gamma$  that occur in this proof. We have therefore obtained a proof of  $\mathcal{Q}$  from  $\Gamma_0$ , so  $\Gamma_0 \vdash \mathcal{Q}$  with  $\Gamma_0$  finite.  $\square$

### 3.2. Compactness Theorem

The corresponding result for satisfaction is the Compactness Theorem for first-order logic, similar to the one encountered for true/false logic in the chapter “Compactness Theorem” in Part One.

**Theorem 7** (Compactness Theorem).

*$\Gamma$  is satisfiable iff every finite subset  $\Gamma_0 \subset \Gamma$  is satisfiable.*

Restatement:  $\Gamma \models \mathcal{Q}$  iff  $\Gamma_0 \models \mathcal{Q}$  for every finite subset  $\Gamma_0 \subset \Gamma$ .

Note: in each case, the forward implication  $\Rightarrow$  is immediate.

For theories, the interesting implication can be restated as follows:

**Theorem 8.**

*If every finite subset of a theory  $\mathcal{T}$  has a model, then  $\mathcal{T}$  itself has a model.*

It is worth taking the time to reflect on how profound such a theorem is. Every finite subset has a model, but of course this model may be different for each  $\Gamma_0$ . Nevertheless, in the end we find a model common to all the formulas.

*Proof of the Compactness Theorem.* We have already said that the forward implication  $\Rightarrow$  is immediate. We will prove the reverse implication  $\Leftarrow$  by contraposition: “if  $\Gamma$  is unsatisfiable, then there is a finite  $\Gamma_0$  that is unsatisfiable.”

$\Gamma$  is unsatisfiable  
 therefore  $\Gamma \models \perp$  since  $\Gamma$  is never satisfied anyway  
 therefore  $\Gamma \vdash \perp$  by the Completeness Theorem, so  $\Gamma$  is inconsistent  
 therefore  $\Gamma_0 \vdash \perp$  for some finite  $\Gamma_0 \subset \Gamma$ , by the finiteness theorem  
 therefore  $\Gamma_0 \models \perp$  by the Soundness Theorem  
 therefore  $\Gamma_0$  is unsatisfiable  
 therefore there exists a finite, unsatisfiable  $\Gamma_0$   
 therefore not (every finite  $\Gamma_0$  is satisfiable)

□

## 4. Closure theorem

The closure theorem explains how to pass from a statement of the Completeness Theorem for closed formulas to a statement for arbitrary formulas.

### 4.1. Statement

**Theorem 9** (Theorem A).

Let  $\mathcal{T}$  be a theory and  $\mathcal{Q}$  a closed formula.

If  $\mathcal{T} \models \mathcal{Q}$  then  $\mathcal{T} \vdash \mathcal{Q}$ .

**Theorem 10** (Theorem C).

Now let  $\Gamma$  be an arbitrary set of formulas and  $\mathcal{Q}$  an arbitrary formula.

If  $\Gamma \models \mathcal{Q}$  then  $\Gamma \vdash \mathcal{Q}$ .

**Theorem 11** (Closure theorem).

Theorem A implies Theorem C.

In other words, to prove the Completeness Theorem in full generality, it is enough to prove it for closed formulas.

### 4.2. Closure

Before proving Theorem C, let us introduce a notion of closure by adding new constants.

**Definition.**

Let  $\mathcal{P}$  be a formula in a language  $\mathcal{L}$  whose free variables are denoted by  $x_1, \dots, x_n$ . Let  $\mathcal{L}^c$  denote the language obtained from  $\mathcal{L}$  by adding  $n$  new constants  $c_1, \dots, c_n$  (these constants must not occur in  $\mathcal{L}$ ). Let  $\mathcal{P}^c$  denote the **substitution of the new constants for the free variables** in  $\mathcal{P}$ , that is, the formula obtained from  $\mathcal{P}$  by replacing each free variable  $x_i$  with the new constant  $c_i$ .

By construction,  $\mathcal{P}^c$  is a closed formula (in the language  $\mathcal{L}^c$ ).

For a set of formulas  $\Gamma$  and a formula  $\mathcal{Q}$ , let  $x_1, \dots, x_n$  denote the set of free variables in  $\Gamma \cup \{\mathcal{Q}\}$ . In this way, we construct a language  $\mathcal{L}^c$  and closed formulas  $\Gamma^c$  (the closure of the formulas in  $\Gamma$  by the new constants) and  $\mathcal{Q}^c$ .

Here are some useful properties for the proof, which are essentially manipulations of notation.

- Point a.

$$\Gamma \models \mathcal{Q} \quad \text{implies} \quad \Gamma^c \models \mathcal{Q}^c$$

Let us justify this in detail. Consider a structure  $\mathcal{S}^c$  (for the language  $\mathcal{L}^c$ ) that satisfies  $\Gamma^c$ . Let  $d_i = c_i^{\mathcal{S}^c}$  denote the interpretation of the constant  $c_i$  in  $\mathcal{S}^c$ .

We can construct a structure  $\mathcal{S}$  (for the language  $\mathcal{L}$ ) by forgetting the new constants  $c_1, \dots, c_n$ , while at the same time constructing an assignment  $s$  by setting  $s(x_i) = d_i$  for each free variable  $x_i$ . For a formula  $\mathcal{P}$  (in  $\Gamma$  or  $\mathcal{Q}$ ), we then have:

$$\models_{\mathcal{S}^c} \mathcal{P}^c \quad \text{iff} \quad \models_{\mathcal{S}, s} \mathcal{P}$$

(This follows directly from substituting the constant  $c_i$  for each free variable  $x_i$ .)

Since  $\models_{\mathcal{S}^c} \Gamma^c$ , we have  $\models_{\mathcal{S}, s} \Gamma$  (by the preceding substitution). By assumption,  $\Gamma \models \mathcal{Q}$ , so  $\models_{\mathcal{S}, s} \mathcal{Q}$ , and hence, going back,  $\models_{\mathcal{S}^c} \mathcal{Q}^c$ .

- Point b. The new constants lemma:

$$\Gamma^c \vdash \mathcal{Q}^c \quad \text{implies} \quad \Gamma \vdash \mathcal{Q}$$

Indeed, suppose we have a natural-deduction proof of  $\mathcal{Q}^c$  from  $\Gamma^c$ . Copy this proof, replacing every occurrence of the new constants  $c_i$  with the variables  $x_i$  on every line. This gives a proof of  $\mathcal{Q}$  from  $\Gamma$ .

| Proof of $\Gamma^c \vdash \mathcal{Q}^c$ |                   |                        | Proof of $\Gamma \vdash \mathcal{Q}$ |                   |                       |
|------------------------------------------|-------------------|------------------------|--------------------------------------|-------------------|-----------------------|
| $\vdash$                                 | $\mathcal{P}_1^c$ | Assumptions $\Gamma^c$ | $\vdash$                             | $\mathcal{P}_1$   | Assumptions $\Gamma$  |
|                                          | $\vdots$          |                        |                                      | $\vdots$          |                       |
|                                          | $\mathcal{P}_l^c$ |                        |                                      | $\mathcal{P}_l$   |                       |
|                                          | $\vdots$          | Proof                  |                                      | $\vdots$          | Proof                 |
|                                          | $\dots c_i \dots$ | The new constants      |                                      | $\dots x_i \dots$ | Back to the variables |
|                                          | $\vdots$          |                        |                                      | $\vdots$          |                       |
|                                          | $\mathcal{Q}^c$   | Conclusion             |                                      | $\mathcal{Q}$     | Conclusion            |

### 4.3. Proof of Theorem C from Theorem A

$$\begin{array}{lll}
 & \Gamma \models \mathcal{Q} & \\
 \text{therefore} & \Gamma^c \models \mathcal{Q}^c & \text{by point a} \\
 \text{therefore} & \Gamma^c \vdash \mathcal{Q}^c & \text{by Theorem A for closed formulas} \\
 \text{therefore} & \Gamma \vdash \mathcal{Q} & \text{by point b}
 \end{array}$$

### 4.4. Universal closure

There is another notion of closure, called *universal closure*.

**Definition.**

Let  $\mathcal{P}$  be a formula whose free variables are denoted by  $x_1, \dots, x_n$ . The *universal closure* of  $\mathcal{P}$  is denoted by  $\overline{\mathcal{P}}$  (or also  $\forall \mathcal{P}$ ):

$$\overline{\mathcal{P}} : \quad \forall x_1 \forall x_2 \dots \forall x_n \mathcal{P}$$

## Exercises

### Completeness Theorem

**Exercise 1.** Let  $\Gamma$  be a set of formulas and  $\mathcal{P}, \mathcal{Q}$  formulas such that:

$$\Gamma \vdash \mathcal{P} \quad \text{and} \quad \mathcal{P} \vdash \mathcal{Q} \quad (*)$$

The goal of the exercise is to obtain  $\Gamma \vdash \mathcal{Q}$ .

1. *First method.* Starting from a natural-deduction proof of  $\Gamma \vdash \mathcal{P}$  and a natural-deduction proof of  $\mathcal{P} \vdash \mathcal{Q}$ , write a proof of  $\Gamma \vdash \mathcal{Q}$ .
2. *Second method.*
  - (a) What statements follow from our assumptions  $(*)$  by applying the Soundness Theorem?
  - (b) Under the assumptions  $(*)$ , show that  $\Gamma \models \mathcal{Q}$ .
  - (c) Conclude.

**Exercise 2.** Let  $\mathcal{L}$  be a language with a binary predicate  $R(x, y)$ . Consider the following three formulas:

1. symmetry  $\mathcal{S} : \forall x \forall y (R(x, y) \rightarrow R(y, x))$
2. transitivity  $\mathcal{T} : \forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z))$
3. reflexivity  $\mathcal{Q} : \forall x R(x, x)$

A relation  $R$  for which these three formulas are true is called an *equivalence relation*. We ask whether the two axioms  $\mathcal{S}$  and  $\mathcal{T}$  alone are sufficient; that is, do we have  $\mathcal{S}, \mathcal{T} \vdash \mathcal{Q}$ ?

1. A student proposes the following proof:

*Let  $x$  be an arbitrary element. Choose an element  $y$  such that  $R(x, y)$ . By axiom  $\mathcal{S}$  (symmetry), we also have  $R(y, x)$ . Applying axiom  $\mathcal{T}$  (transitivity) to  $R(x, y)$  and  $R(y, x)$  gives  $R(x, x)$ . The relation is therefore reflexive, and  $\mathcal{Q}$  is proved.*

What is the fundamental logical error in this reasoning?

2. Find a structure  $\mathcal{S}$ , that is, define a domain  $E$  and specify the relation  $R$  on it, such that the formulas  $\mathcal{S}$  and  $\mathcal{T}$  are true in it, but  $\mathcal{Q}$  is not.
3. Do we have  $\mathcal{S}, \mathcal{T} \vdash \mathcal{Q}$ ?

### Applications to theories

**Exercise 3.** Show that the following two definitions of an *inconsistent* theory  $\mathcal{T}$  are equivalent:

- (i) there is a  $\mathcal{Q}$  such that  $\mathcal{T} \vdash \mathcal{Q}$  and  $\mathcal{T} \vdash \neg \mathcal{Q}$ ,
- (ii)  $\mathcal{T} \vdash \perp$ .

**Exercise 4.** Using the Completeness Theorem “if  $\mathcal{T} \models \mathcal{Q}$ , then  $\mathcal{T} \vdash \mathcal{Q}$ ”, prove that:  
“if  $\mathcal{T}$  is unsatisfiable, then  $\mathcal{T}$  is inconsistent”.

Then prove the converse.

**Exercise 5.** Recall Theorems A and B:

- Theorem A.

$$\mathcal{T} \models \mathcal{Q} \quad \text{iff} \quad \mathcal{T} \vdash \mathcal{Q}.$$

- Theorem B.

$$\mathcal{T} \text{ is consistent} \quad \text{iff} \quad \mathcal{T} \text{ has a model.}$$

Assume Theorem B. Deduce Theorem A.

### Application to finiteness

**Exercise 6.** Let  $\mathcal{L}$  be a language (containing the equality symbol  $=$ ). For every integer  $n \geq 1$ , let  $\mathcal{P}_n$  denote the formula expressing “there are at least  $n$  distinct elements”.

1. Write the formulas  $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$  explicitly.
2. Let  $\mathcal{T}$  be a theory with finite models of arbitrarily large size; that is, for every integer  $k$ ,  $\mathcal{T}$  has a model with at least  $k$  elements.  
Let  $\Gamma = \mathcal{T} \cup \{\mathcal{P}_n \mid n \geq 1\}$ . Show that every finite subset of  $\Gamma$  has a model.
3. Using the Compactness Theorem, deduce that  $\mathcal{T}$  has an infinite model.
4. Conclusion: is there a first-order theory  $\mathcal{T}_{\text{finite}}$  whose models are **exactly** all finite structures?

### Closure theorem

**Exercise 7.** For a formula  $\mathcal{P}$  whose free variables are denoted by  $x_1, \dots, x_n$ , its *existential closure* is the formula  $\mathcal{P}_{\exists}$  (also denoted by  $\exists \mathcal{P}$ ):

$$\mathcal{P}_{\exists} : \quad \exists x_1 \dots \exists x_n \mathcal{P}$$

1. Show that  $\mathcal{P}$  is satisfiable if and only if  $\mathcal{P}_{\exists}$  has a model.
2. Prove the equivalence  $\neg(\overline{\mathcal{P}}) \equiv (\neg \mathcal{P})_{\exists}$ .  
Recall that  $\overline{\mathcal{P}}$  denotes the universal closure and that  $\mathcal{P} \equiv \mathcal{Q}$  means  $(\mathcal{P} \models \mathcal{Q} \text{ and } \mathcal{Q} \models \mathcal{P})$ ; see the chapter “Satisfaction”.
3. Show that if  $\Gamma \vdash \mathcal{Q}$ , then  $\Gamma \vdash \mathcal{Q}_{\exists}$ . Is the converse true?

# Proof of the Completeness Theorem

## Chapter 11

*We prove the Completeness Theorem.*

## 1. The Completeness Theorem

### 1.1. Statement

Recall the statement that we studied carefully in the chapter “Completeness Theorem” but have not yet proved.

**Theorem 1** (Completeness Theorem).

*If  $\Gamma \models \mathcal{Q}$ , then  $\Gamma \vdash \mathcal{Q}$ .*

In other words, everything that is true is provable.

### 1.2. To prove: “If $\mathcal{T}$ is consistent, then $\mathcal{T}$ has a model.”

By the closure theorem from the previous chapter, the proof of the Completeness Theorem in the general setting reduces to the setting of closed formulas. Let  $\mathcal{T}$  be a theory and  $\mathcal{Q}$  a closed formula.

**Theorem 2** (Theorem A).

*If  $\mathcal{T} \models \mathcal{Q}$ , then  $\mathcal{T} \vdash \mathcal{Q}$ .*

We will therefore study the case of theories. Theorem A will be obtained from the following equivalent Theorem B:

**Theorem 3** (Theorem B).

*If  $\mathcal{T}$  is consistent, then  $\mathcal{T}$  has a model.*

Thus, we will have the chain:

$$\text{Theorem B} \implies \text{Theorem A} \implies \text{Completeness Theorem}$$

Theorem B is therefore what we will prove in this chapter, after explaining how to deduce Theorem A from it.

### 1.3. Review of terminology

- **Theory**: a set  $\mathcal{T}$  of closed formulas.
- $\mathcal{T}$  is **inconsistent** if  $\mathcal{T} \vdash \perp$ ; otherwise,  $\mathcal{T}$  is **consistent**.
- $\mathcal{T}$  has a **model** if there is a structure  $S$  such that  $\models_S \mathcal{T}$ ; that is, a structure in which all the formulas of  $\mathcal{T}$  become true.

In the previous chapter, we saw the following proposition:

**Proposition 1.**

$\mathcal{T} \vdash Q$  iff  $\mathcal{T} \cup \{\neg Q\}$  is inconsistent.

Here is how to prove Theorem A from Theorem B (we saw the converse in the previous chapter).

*Proof.* We will prove Theorem A in its contrapositive form: if  $\mathcal{T} \not\vdash Q$ , then  $\mathcal{T} \not\models Q$ .

We will use negations of operators sparingly:  $\mathcal{T} \not\vdash Q$  means  $\text{not}(\mathcal{T} \vdash Q)$  and  $\mathcal{T} \not\models Q$  means  $\text{not}(\mathcal{T} \models Q)$ .

Assumption:  $\text{not}(\mathcal{T} \vdash Q)$ . Goal:  $\text{not}(\mathcal{T} \models Q)$ .

$\text{not}(\mathcal{T} \vdash Q)$

therefore  $\mathcal{T} \cup \{\neg Q\}$  is consistent      by Proposition 1 above

therefore  $\mathcal{T} \cup \{\neg Q\}$  has a model      by Theorem B

therefore  $\models_{\mathcal{S}} \mathcal{T} \cup \{\neg Q\}$       for some structure  $\mathcal{S}$

therefore  $\models_{\mathcal{S}} \mathcal{T}$  and  $\models_{\mathcal{S}} \neg Q$       since  $\mathcal{T}$  and  $\neg Q$  are both true in  $\mathcal{S}$

therefore  $\text{not}(\mathcal{T} \models Q)$       since  $\mathcal{T}$  is true and  $Q$  is false in  $\mathcal{S}$

□

## 2. Complete theory

### 2.1. Definition

**Definition.**

A theory  $\mathcal{T}$  is said to be *complete* if:

- $\mathcal{T}$  is consistent,
- and, for every closed formula  $Q$ , either  $\mathcal{T} \vdash Q$  or  $\mathcal{T} \vdash \neg Q$ .

Thus, in a complete theory, for every closed formula  $Q$ , either it or its negation can be proved, but not both at the same time. The name “complete” is therefore well chosen.

### 2.2. Satisfaction

Here is an important consequence when working with a complete theory: formulas are true or false independently of the structure.

**Proposition 2.**

Let  $\mathcal{T}$  be a complete theory and  $Q$  a closed formula. Let  $\mathcal{S}_1$  and  $\mathcal{S}_2$  be two models of  $\mathcal{T}$ . Then:

$$\models_{\mathcal{S}_1} Q \quad \text{iff} \quad \models_{\mathcal{S}_2} Q.$$

*Proof.* Since  $\mathcal{T}$  is complete, consider the two cases: either  $\mathcal{T} \vdash Q$  or  $\mathcal{T} \vdash \neg Q$ .

- If  $\mathcal{T} \vdash Q$ , then  $\mathcal{T} \models Q$  by the Soundness Theorem, so  $\models_{\mathcal{S}_1} Q$  and  $\models_{\mathcal{S}_2} Q$  because  $\mathcal{S}_1$  and  $\mathcal{S}_2$  are models of  $\mathcal{T}$ ; that is,  $Q$  is true in both  $\mathcal{S}_1$  and  $\mathcal{S}_2$ .
- If  $\mathcal{T} \vdash \neg Q$ , the same reasoning gives  $\models_{\mathcal{S}_1} \neg Q$  and  $\models_{\mathcal{S}_2} \neg Q$ ; that is,  $Q$  is false in both  $\mathcal{S}_1$  and  $\mathcal{S}_2$ .

□

## 2.3. A lemma

In a complete theory, there are no traps. We will prove the following statements, which may seem obvious but require this notion.

### Lemma 1.

Let  $\mathcal{T}$  be a complete theory.

1.  $\mathcal{T} \vdash \neg \mathcal{P}$  iff  $\text{not}(\mathcal{T} \vdash \mathcal{P})$
2.  $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$  iff  $(\mathcal{T} \vdash \mathcal{P} \text{ and } \mathcal{T} \vdash \mathcal{Q})$
3.  $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$  iff  $(\mathcal{T} \vdash \mathcal{P} \text{ or } \mathcal{T} \vdash \mathcal{Q})$

*Proof.*

1.
  - Suppose  $\mathcal{T} \vdash \neg \mathcal{P}$ . By contradiction, if  $\mathcal{T} \vdash \mathcal{P}$ , then  $\mathcal{T} \vdash (\mathcal{P} \wedge \neg \mathcal{P})$ , so  $\mathcal{T} \vdash \perp$ . This contradicts the consistency of  $\mathcal{T}$  (since  $\mathcal{T}$  is complete). Therefore,  $\text{not}(\mathcal{T} \vdash \mathcal{P})$ .
  - Suppose  $\text{not}(\mathcal{T} \vdash \mathcal{P})$ . Since  $\mathcal{T}$  is complete ( $\mathcal{T} \vdash \mathcal{P}$  or  $\mathcal{T} \vdash \neg \mathcal{P}$ ), we therefore have  $\mathcal{T} \vdash \neg \mathcal{P}$ .
2.
  - If  $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$ , then  $\mathcal{T} \vdash \mathcal{P}$  and  $\mathcal{T} \vdash \mathcal{Q}$  because  $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$  and  $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}$ .
  - If  $\mathcal{T} \vdash \mathcal{P}$  and  $\mathcal{T} \vdash \mathcal{Q}$ , then  $\mathcal{T} \vdash \mathcal{P} \wedge \mathcal{Q}$ , because  $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$ .

Note: this point does not use the assumption that  $\mathcal{T}$  is complete.

3.
  - If  $(\mathcal{T} \vdash \mathcal{P} \text{ or } \mathcal{T} \vdash \mathcal{Q})$ , then  $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$  because  $\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q}$  and  $\mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}$ . (There is no need for  $\mathcal{T}$  to be complete.)
  - Suppose  $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$ . Suppose, for contradiction, that  $\mathcal{T} \not\vdash \mathcal{P}$  and  $\mathcal{T} \not\vdash \mathcal{Q}$ . Since  $\mathcal{T} \not\vdash \mathcal{P}$ , we have  $\mathcal{T} \vdash \neg \mathcal{P}$  because  $\mathcal{T}$  is complete (by point 1); similarly, since  $\mathcal{T} \not\vdash \mathcal{Q}$ , we have  $\mathcal{T} \vdash \neg \mathcal{Q}$ . Thus, by point 2,  $\mathcal{T} \vdash \neg \mathcal{P} \wedge \neg \mathcal{Q}$ , so  $\mathcal{T} \vdash \neg(\mathcal{P} \vee \mathcal{Q})$ . But then we have both  $\mathcal{T} \vdash \mathcal{P} \vee \mathcal{Q}$  (by assumption) and  $\mathcal{T} \vdash \neg(\mathcal{P} \vee \mathcal{Q})$ , so  $\mathcal{T}$  is inconsistent. This is a contradiction because  $\mathcal{T}$  is complete and hence consistent.

□

## 3. Constructing a model from a complete theory

We begin the proof of the Completeness Theorem in the form of Theorem B: “if  $\mathcal{T}$  is consistent, then  $\mathcal{T}$  has a model”. We begin this proof somewhat from the end. We will start from a complete theory with additional assumptions and construct a model. We will explain later how to obtain these additional assumptions. There are several proofs of the Completeness Theorem; here we adopt the one based on Henkin witnesses.

### 3.1. Strengthened assumptions

Let  $\mathcal{L}$  be a language and  $\mathcal{T}$  a theory. We assume the following properties of  $\mathcal{T}$ :

- (i)  $\mathcal{T}$  is consistent.
- (ii)  $\mathcal{T}$  is complete.
- (iii)  $\mathcal{T}$  has Henkin witnesses.

Note: (ii)  $\implies$  (i), but we emphasize that in the end we would like (i) to be the only assumption.

**Definition.**

$\mathcal{T}$  has **Henkin witnesses** if, for every formula  $\mathcal{P}$  in  $\mathcal{L}$  with a single free variable  $x$ , there is a constant  $c_{\mathcal{P}}$  in  $\mathcal{L}$  such that the following formula:

$$\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})$$

belongs to  $\mathcal{T}$ .

Notes.

- We could also write: there exists  $c \in \mathcal{L}$  such that:

$$(\exists x \mathcal{P}(x)) \rightarrow \mathcal{P}(c) \in \mathcal{T},$$

if we do not wish to emphasize the dependence of  $c$  on  $\mathcal{P}$ .

- With Henkin witnesses, a formula “ $\exists x \mathcal{P}(x)$ ” is no longer an abstract formula, because we can obtain a concrete realization “ $\mathcal{P}(c_{\mathcal{P}})$ ”, hence the name “witness”.

Here are two more additional assumptions, this time concerning the language  $\mathcal{L}$ :

- (iv)  $\mathcal{L}$  is a countable language (the set of symbols in  $\mathcal{L}$  is countable). This implies that the set of formulas in  $\mathcal{L}$  is also countable (see the chapter “Compactness Theorem”).
- (v)  $\mathcal{L}$  does not contain the equality predicate “ $=$ ”. Without this assumption, the proof would become more technical because we would have to work with equivalence classes for the relation defined by  $t_1 \sim t_2$  whenever  $t_1 = t_2$  holds.

Despite these simplifying assumptions, the proof remains difficult.

### 3.2. Constructing the model

For a theory  $\mathcal{T}$  satisfying assumptions (i), (ii), and (iii) in a language  $\mathcal{L}$  satisfying assumptions (iv) and (v), we will construct a model  $\mathcal{S}$  of  $\mathcal{T}$ ; that is, a structure  $\mathcal{S}$  such that  $\models_{\mathcal{S}} \mathcal{T}$ .

Constructing the structure  $\mathcal{S}$  means specifying:

- a domain  $E$ ,
- an interpretation  $c^{\mathcal{S}}$  of each constant  $c$  in  $\mathcal{L}$ ,
- an interpretation  $f^{\mathcal{S}}$  of each function  $f$  in  $\mathcal{L}$ ,
- an interpretation  $P^{\mathcal{S}}$  of each predicate  $P$  in  $\mathcal{L}$ .

We will then have to prove that every formula  $\mathcal{P}$  in  $\mathcal{T}$  is true when interpreted in  $\mathcal{S}$ .

Here we go! The construction is more or less automatic. We define the objects so that they have the desired properties. Here is the structure  $\mathcal{S}$ :

- The domain  $E$  is the set of closed terms in  $\mathcal{L}$ . Recall that terms are formed from constants and variables, to which functions can be applied recursively. Here, the closed terms are therefore only constants and iterated applications of functions to constants. Examples of closed terms:  $c_1$ ,  $c_2$ ,  $f(c_1)$ ,  $g(c_1, c_2)$ ,  $f(f(c_1))$ ,  $g(c_1, f(c_2))$ , etc.

Note: if equality were part of the language,  $E$  would have to be the set of equivalence classes of closed terms modulo the equivalence relation  $t_1 \sim t_2$  iff  $\mathcal{T} \vdash (t_1 = t_2)$ .

- If  $c$  is a constant symbol in  $\mathcal{L}$ , then  $c$  is a closed term and hence an element of  $E$ . We simply define  $c^{\mathcal{S}} = c$ .

- Let  $f$  be an  $n$ -ary function symbol in  $\mathcal{L}$ . Defining  $f^S$  means defining an “actual” function  $f^S : E^n \rightarrow E$ . Let  $t_1, \dots, t_n$  be elements of  $E$ . They are closed terms in  $\mathcal{L}$ , so  $f(t_1, \dots, t_n)$  is also a closed term and hence an element of  $E$ . Set  $f^S(t_1, \dots, t_n) = f(t_1, \dots, t_n)$ .
- Let  $P$  be an  $n$ -place predicate in  $\mathcal{L}$ . Defining the interpretation  $P^S$  means defining a function  $P^S : E^n \rightarrow \{T, F\}$ . Here is how to assign a truth value to each element  $(t_1, \dots, t_n) \in E^n$ : set  $P^S(t_1, \dots, t_n) = T$  if  $\mathcal{T} \vdash P(t_1, \dots, t_n)$ , and set  $P^S(t_1, \dots, t_n) = F$  otherwise.

Thus, by construction:  $\models_S P(t_1, \dots, t_n)$  iff  $\mathcal{T} \vdash P(t_1, \dots, t_n)$ . We will later generalize this statement to arbitrary formulas.

### 3.3. Satisfaction

It remains to show that our structure  $S$  is indeed a model of  $\mathcal{T}$ ; that is,  $\models_S \mathcal{T}$ . The proof is based on the following result:

**Lemma 2.**

Let  $\mathcal{Q}$  be a closed formula.

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{iff} \quad \models_S \mathcal{Q}$$

Assume this lemma for a moment and let us show that  $S$  is a model of  $\mathcal{T}$ . Indeed, let  $\mathcal{P}$  be a formula in  $\mathcal{T}$ . Of course,  $\mathcal{T} \vdash \mathcal{P}$ , so the lemma gives  $\models_S \mathcal{P}$ . Since this is true for every formula in  $\mathcal{T}$ , we indeed have  $\models_S \mathcal{T}$ .

*Proof of the lemma.* For simplicity, and without loss of generality, assume that the formula  $\mathcal{Q}$  contains only the connectives  $\neg$  and  $\wedge$ , together with the quantifier  $\exists$ . For the connectives, see the chapter “Which Connectives Are Useful?” for the reduction to these two connectives. A formula “ $\forall x \mathcal{P}(x)$ ” is equivalent to “ $\neg(\exists x \neg \mathcal{P}(x))$ ”, which allows us to turn every  $\forall$  quantifier into an  $\exists$  quantifier (with additional negations).

Let us prove the result by induction on the construction of the formula  $\mathcal{Q}$ .

- Case where  $\mathcal{Q}$  is an atomic formula.

Then  $\mathcal{Q} = P(t_1, \dots, t_n)$ , where  $P$  is a predicate and  $t_1, \dots, t_n$  are closed terms. By construction of  $S$ ,

$$\mathcal{T} \vdash P(t_1, \dots, t_n) \quad \text{iff} \quad \models_S P(t_1, \dots, t_n),$$

so the lemma is true in this case.

- Case where  $\mathcal{Q} = \neg \mathcal{P}$ .

By induction, assume the lemma is true for  $\mathcal{P}$ , and prove it for  $\mathcal{Q} = \neg \mathcal{P}$ .

$$\begin{aligned}
 & \mathcal{T} \vdash \mathcal{Q} \\
 \text{iff} & \quad \mathcal{T} \vdash \neg \mathcal{P} \\
 \text{iff} & \quad \text{not}(\mathcal{T} \vdash \mathcal{P}) \quad \text{by Lemma 1, since } \mathcal{T} \text{ is complete} \\
 \text{iff} & \quad \text{not}(\models_S \mathcal{P}) \quad \text{by the induction hypothesis} \\
 \text{iff} & \quad \models_S \neg \mathcal{P} \\
 \text{iff} & \quad \models_S \mathcal{Q}
 \end{aligned}$$

The key step is the transition from the second line to the third, which uses assumption (ii), that  $\mathcal{T}$  is complete, through Lemma 1.

- Case where  $Q = \mathcal{P}_1 \wedge \mathcal{P}_2$ .

By induction, assume the lemma is true for  $\mathcal{P}_1$  and  $\mathcal{P}_2$ .

$$\begin{aligned}
 & \mathcal{T} \vdash Q \\
 \text{iff} \quad & \mathcal{T} \vdash \mathcal{P}_1 \wedge \mathcal{P}_2 \\
 \text{iff} \quad & (\mathcal{T} \vdash \mathcal{P}_1 \text{ and } \mathcal{T} \vdash \mathcal{P}_2) \quad \text{by Lemma 1} \\
 \text{iff} \quad & (\models_{\mathcal{S}} \mathcal{P}_1 \text{ and } \models_{\mathcal{S}} \mathcal{P}_2) \quad \text{by the induction hypothesis} \\
 \text{iff} \quad & \models_{\mathcal{S}} \mathcal{P}_1 \wedge \mathcal{P}_2 \\
 \text{iff} \quad & \models_{\mathcal{S}} Q
 \end{aligned}$$

- Case where  $Q = \exists x \mathcal{P}(x)$ .

Assume the lemma is true for  $\mathcal{P}$ .

- If  $\mathcal{T} \vdash Q$ , that is, if  $\mathcal{T} \vdash \exists x \mathcal{P}(x)$ , then assumption (iii) on Henkin witnesses tells us that there is a  $c_{\mathcal{P}} \in \mathcal{L}$  such that “ $(\exists x \mathcal{P}(x)) \rightarrow \mathcal{P}(c_{\mathcal{P}})$ ” is a formula in  $\mathcal{T}$ . Since  $\mathcal{T} \vdash \exists x \mathcal{P}(x)$  and  $\mathcal{T} \vdash [(\exists x \mathcal{P}(x)) \rightarrow \mathcal{P}(c_{\mathcal{P}})]$ , we have  $\mathcal{T} \vdash \mathcal{P}(c_{\mathcal{P}})$ . Therefore, by the induction hypothesis,  $\models_{\mathcal{S}} \mathcal{P}(c_{\mathcal{P}})$ , and by the definition of satisfaction for  $\exists$ , we deduce  $\models_{\mathcal{S}} \exists x \mathcal{P}(x)$ ; that is,  $\models_{\mathcal{S}} Q$ .
- If  $\models_{\mathcal{S}} Q$ , then  $\models_{\mathcal{S}} \exists x \mathcal{P}(x)$ , so there is a closed term  $t$ , whose interpretation is  $t^{\mathcal{S}} \in E$ , such that  $\models_{\mathcal{S}} \mathcal{P}(t)$ . By the induction hypothesis,  $\mathcal{T} \vdash \mathcal{P}(t)$ . Thus, by the  $\exists$  introduction rule,  $\mathcal{T} \vdash \exists x \mathcal{P}(x)$ , and hence  $\mathcal{T} \vdash Q$ .

□

## 4. Constructing a complete theory

Let us explain how to make a theory complete by adding formulas.

### 4.1. Construction

Let  $\mathcal{T}$  be a consistent theory in a language  $\mathcal{L}$ . We are going to construct a theory  $\mathcal{T}'$ , in the same language  $\mathcal{L}$ , such that  $\mathcal{T} \subset \mathcal{T}'$  and  $\mathcal{T}'$  is complete. We assume that the language  $\mathcal{L}$  is countable, so the set of formulas is countable as well. Consider an enumeration  $\mathcal{P}_0, \mathcal{P}_1, \dots$  of all the closed formulas of  $\mathcal{L}$ . We construct  $\mathcal{T}'$  as the limit of  $(\mathcal{T}_n)_{n \geq 0}$ , defined recursively as follows:

- $\mathcal{T}_0 = \mathcal{T}$
- - $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$  if this union is consistent,
  - $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg \mathcal{P}_n\}$  otherwise.

Finally, let  $\mathcal{T}' = \bigcup_{n \geq 0} \mathcal{T}_n$ .

#### Proposition 3.

Each  $\mathcal{T}_n$  is consistent, and  $\mathcal{T}'$  is complete.

### 4.2. Proof

*Proof.*

- Let us prove by induction that the  $\mathcal{T}_n$  are consistent. This is true for  $\mathcal{T}_0 = \mathcal{T}$  by assumption. Suppose it is true for  $\mathcal{T}_n$ . In the first case,  $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$  is consistent by construction. In the second case,  $\mathcal{T}_n \cup \{\mathcal{P}_n\}$  is not consistent, so  $\mathcal{T}_n \vdash \neg \mathcal{P}_n$  (Proposition 1). If  $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg \mathcal{P}_n\}$  were not consistent either, then  $\mathcal{T}_n \vdash \neg(\neg \mathcal{P}_n)$ , and hence  $\mathcal{T}_n \vdash \mathcal{P}_n$ . We would then have  $\mathcal{T}_n \vdash \neg \mathcal{P}_n$  and  $\mathcal{T}_n \vdash \mathcal{P}_n$ , contradicting the induction hypothesis that  $\mathcal{T}_n$  is consistent. Thus, in every case,  $\mathcal{T}_{n+1}$  is consistent.
- $\mathcal{T}'$  is consistent. Indeed, otherwise we would have  $\mathcal{T}' \vdash \perp$ . Since a proof consists of finitely many steps, and therefore involves finitely many formulas, these formulas are contained in some  $\mathcal{T}_n$  for sufficiently large  $n$ . We would then have  $\mathcal{T}_n \vdash \perp$ , contradicting the consistency of  $\mathcal{T}_n$ .
- $\mathcal{T}'$  is complete. Indeed, let  $\mathcal{Q}$  be a closed formula of  $\mathcal{L}$ . It is one of the  $\mathcal{P}_n$ . Therefore,  $\mathcal{P}_n \in \mathcal{T}_{n+1}$  or  $\neg \mathcal{P}_n \in \mathcal{T}_{n+1}$ . Thus, either  $\mathcal{Q}$  or  $\neg \mathcal{Q}$  is an element of  $\mathcal{T}'$ . Consequently,  $\mathcal{T}' \vdash \mathcal{Q}$  or  $\mathcal{T}' \vdash \neg \mathcal{Q}$ .

□

A useful observation for what follows: if  $\mathcal{T}$  has Henkin witnesses, then so does  $\mathcal{T}'$ . Indeed, “ $(\exists x \mathcal{P}(x)) \rightarrow \mathcal{P}(c_{\mathcal{P}})$ ” belongs to  $\mathcal{T}$  and therefore to  $\mathcal{T}'$ .

## 5. Constructing witnesses

Start with a theory  $\mathcal{T}$  in a language  $\mathcal{L}$ . We will extend the language in order to obtain a new theory  $\mathcal{T}'$  in a language  $\mathcal{L}'$  that has Henkin witnesses; that is:

$$“\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})” \text{ is a formula in } \mathcal{T}'.$$

In addition,  $\mathcal{T}'$  must be consistent, just like  $\mathcal{T}$ .

### 5.1. Extending the language and the theory

$\mathcal{L}'$  will be constructed from  $\mathcal{L}$  by adding new constant symbols  $c_0, c_1, \dots$ . Before giving the full proof, here is the fundamental step.

Let  $\mathcal{T}$  be a consistent theory in a language  $\mathcal{L}$ . Let  $\mathcal{P}$  be a formula of  $\mathcal{L}$  whose only free variable is  $x$ . Let  $c_{\mathcal{P}}$  be a new constant symbol (one that does not belong to  $\mathcal{L}$ ). Let  $\mathcal{L}' = \mathcal{L} \cup \{c_{\mathcal{P}}\}$ .

**Claim.**  $\mathcal{T}' = \mathcal{T} \cup \{\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})\}$  is still consistent.

By construction, we have a Henkin witness (just for the formula  $\mathcal{P}$ ), and consistency has been preserved.

*Proof.* Suppose, for a contradiction, that  $\mathcal{T}'$  is not consistent:

$$\begin{aligned} & \mathcal{T} \cup \{\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})\} \vdash \perp \\ \text{therefore} \quad & \mathcal{T} \vdash \neg(\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})) \\ \text{therefore} \quad & \mathcal{T} \vdash \exists x \mathcal{P}(x) \wedge \neg \mathcal{P}(c_{\mathcal{P}}) \quad \text{since } \neg(\mathcal{A} \rightarrow \mathcal{B}) \equiv \mathcal{A} \wedge \neg \mathcal{B} \\ \text{therefore} \quad & \mathcal{T} \vdash \exists x \mathcal{P}(x) \text{ and } \mathcal{T} \vdash \neg \mathcal{P}(c_{\mathcal{P}}) \\ \text{therefore} \quad & \mathcal{T} \vdash \exists x \mathcal{P}(x) \text{ and } \mathcal{T} \vdash \forall x \neg \mathcal{P}(x) \quad \text{by the } \forall \text{ introduction rule} \\ \text{therefore} \quad & \mathcal{T} \vdash \exists x \mathcal{P}(x) \text{ and } \mathcal{T} \vdash \neg(\exists x \mathcal{P}(x)) \quad \text{by the negation of } \forall \\ \text{therefore} \quad & \mathcal{T} \vdash \mathcal{Q} \text{ and } \mathcal{T} \vdash \neg \mathcal{Q} \quad \text{for } \mathcal{Q} = \exists x \mathcal{P}(x) \end{aligned}$$

This contradicts the consistency of  $\mathcal{T}$ . Conclusion:  $\mathcal{T}'$  is consistent. Note the use of  $\forall$  introduction to pass from “ $\neg \mathcal{P}(c_{\mathcal{P}})$ ” to “ $\forall x \neg \mathcal{P}(x)$ ”. This is legitimate because the constant  $c_{\mathcal{P}}$  does not occur in  $\mathcal{T}$ . Indeed, the formulas in  $\mathcal{T}$  are formulas of  $\mathcal{L}$ , which does not contain the constant  $c_{\mathcal{P}}$ . □

One might think that we could do this for every formula  $\mathcal{P}_0, \mathcal{P}_1, \dots$  of  $\mathcal{L}$ , but matters are a little more subtle. Indeed, at each step we enlarge the language, so there are more and more formulas, which complicates the recursion.

## 5.2. Construction

Let  $\mathcal{T}$  be a consistent theory in a language  $\mathcal{L}$ . We will recursively construct a sequence of theories  $(\mathcal{T}_n)_{n \geq 0}$  and a sequence of languages  $(\mathcal{L}_n)_{n \geq 0}$  as follows:

- $\mathcal{L}_0 = \mathcal{L}$  and  $\mathcal{T}_0 = \mathcal{T}$ ,
- and then, for  $n \geq 0$ :
  - $\mathcal{L}_{n+1} = \mathcal{L}_n \cup \{c_{\mathcal{P}}\}_{\mathcal{P}}$ , where the union ranges over all formulas  $\mathcal{P}$  of  $\mathcal{L}_n$  having exactly one free variable,
  - $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})\}_{\mathcal{P}}$ .

Finally, let  $\mathcal{L}' = \bigcup_{n \geq 0} \mathcal{L}_n$  and  $\mathcal{T}' = \bigcup_{n \geq 0} \mathcal{T}_n$ .

**Claim.**  $\mathcal{T}'$  has Henkin witnesses.

*Proof.* Let  $\mathcal{P}$  be a formula of  $\mathcal{L}'$  whose only free variable is  $x$ . It contains finitely many symbols of  $\mathcal{L}'$ , so for sufficiently large  $n$ ,  $\mathcal{P}$  is a formula of  $\mathcal{L}_n$ . Therefore,  $\exists x \mathcal{P}(x) \rightarrow \mathcal{P}(c_{\mathcal{P}})$  is a formula in  $\mathcal{T}_{n+1}$ , and hence in  $\mathcal{T}'$ .  $\square$

**Claim.**  $\mathcal{T}'$  is consistent.

*Proof.* We will show that each  $\mathcal{T}_n$  is consistent. This will prove that  $\mathcal{T}'$  is consistent. Indeed, if we had  $\mathcal{T}' \vdash \perp$ , then a proof of this would only involve a finite number of formulas from  $\mathcal{T}'$ , which are therefore all included in some  $\mathcal{T}_n$  for a sufficiently large  $n$ . For this  $n$ , we have  $\mathcal{T}_n \vdash \perp$ , which contradicts the consistency of  $\mathcal{T}_n$ .

Let us show by induction that  $\mathcal{T}_n$  (for  $n \geq 0$ ) is consistent. For  $n = 0$ ,  $\mathcal{T}_0 = \mathcal{T}$ , so  $\mathcal{T}_0$  is consistent by hypothesis.

Suppose  $\mathcal{T}_n$  is consistent and, for the sake of contradiction, that  $\mathcal{T}_{n+1}$  is not. A proof of  $\mathcal{T}_{n+1} \vdash \perp$  involves only a finite number of formulas. There thus exist a finite number of formulas  $\mathcal{P}_1, \dots, \mathcal{P}_k$  of  $\mathcal{L}_n$  such that, letting  $\mathcal{H}_i$  denote the Henkin axiom “ $\exists x \mathcal{P}_i(x) \rightarrow \mathcal{P}_i(c_{\mathcal{P}_i})$ ”, we have:

$$\mathcal{T}_n \cup \{\mathcal{H}_1, \dots, \mathcal{H}_k\} \vdash \perp$$

Let  $\mathcal{S}_m = \mathcal{T}_n \cup \{\mathcal{H}_1, \dots, \mathcal{H}_m\}$  for  $0 \leq m \leq k$ . Let us show by finite induction on  $m$  that  $\mathcal{S}_m$  is consistent.

- For  $m = 0$ ,  $\mathcal{S}_0 = \mathcal{T}_n$  is consistent by the induction hypothesis.
- Suppose  $\mathcal{S}_{m-1}$  is consistent. The axiom  $\mathcal{H}_m$  is  $\exists x \mathcal{P}_m(x) \rightarrow \mathcal{P}_m(c_{\mathcal{P}_m})$ . By definition,  $c_{\mathcal{P}_m}$  is a new constant symbol that does not belong to  $\mathcal{L}_n$ . Furthermore, since the formulas  $\mathcal{P}_i$  are pairwise distinct, the constant  $c_{\mathcal{P}_m}$  is different from the constants  $c_{\mathcal{P}_1}, \dots, c_{\mathcal{P}_{m-1}}$ . Thus, the constant  $c_{\mathcal{P}_m}$  does not appear anywhere in the theory  $\mathcal{S}_{m-1}$ . According to the fundamental claim proved at the beginning of the section, adding  $\mathcal{H}_m$  to the consistent theory  $\mathcal{S}_{m-1}$  yields a consistent theory. Therefore,  $\mathcal{S}_m = \mathcal{S}_{m-1} \cup \{\mathcal{H}_m\}$  is consistent.

Consequently,  $\mathcal{S}_k = \mathcal{T}_n \cup \{\mathcal{H}_1, \dots, \mathcal{H}_k\}$  is consistent. This contradicts the fact that  $\mathcal{S}_k \vdash \perp$ . Conclusion:  $\mathcal{T}_{n+1}$  is indeed consistent.  $\square$

### 5.3. Complete proof of the theorem

We have given the details of the proof of the Completeness Theorem, but somewhat out of order. Here is the proof plan in the proper order.

- **Assumptions.**  $\mathcal{T}$  is a consistent theory, and  $\mathcal{L}$  is a countable language.
- **Step 1: Add Henkin witnesses.** We refer to the beginning of Section 5. We obtain a theory  $\mathcal{T}'$  in a new language  $\mathcal{L}'$ , with  $\mathcal{T} \subset \mathcal{T}'$  and  $\mathcal{L} \subset \mathcal{L}'$ .  $\mathcal{T}'$  has Henkin witnesses and remains consistent.  $\mathcal{L}'$  is obtained from  $\mathcal{L}$  by adding countably many constants  $c_{\mathcal{P}}$ , so  $\mathcal{L}'$  remains countable.
- **Step 2: Make the theory complete.** See Section 4. Starting with  $\mathcal{T}'$  and  $\mathcal{L}'$ , we obtain a complete theory  $\mathcal{T}''$  with  $\mathcal{T}' \subset \mathcal{T}''$ . Henkin witnesses are also preserved (see the observation at the end of Section 4). The language is unchanged; let  $\mathcal{L}'' = \mathcal{L}'$ .
- **Step 3: Construct the model.** This is Section 3, applied to  $\mathcal{T}''$  and  $\mathcal{L}''$ . We obtain a model  $\mathcal{S}''$  of  $\mathcal{T}''$  in  $\mathcal{L}''$ . This model naturally restricts to a model  $\mathcal{S}$  of  $\mathcal{T}$  in  $\mathcal{L}$ . Indeed,  $\mathcal{L}''$  is obtained from  $\mathcal{L}$  by adding constants  $c_{\mathcal{P}}$ . Consider the structure  $\mathcal{S}$  that is identical to  $\mathcal{S}''$  except that these constants (which do not exist in  $\mathcal{L}$ ) are not interpreted. Let  $\mathcal{P}$  be a formula of  $\mathcal{T}$  in  $\mathcal{L}$ . It is therefore also a formula of  $\mathcal{T}''$  in  $\mathcal{L}''$ . Since  $\mathcal{S}''$  is a model of  $\mathcal{T}''$ , we have  $\models_{\mathcal{S}''} \mathcal{P}$ . But since  $\mathcal{P}$  is a formula of  $\mathcal{L}$ , it contains none of the additional constant symbols, so we also have  $\models_{\mathcal{S}} \mathcal{P}$ . Thus,  $\mathcal{S}$  is a model of  $\mathcal{T}$ .

This completes the proof of Gödel's Completeness Theorem. This theorem was the subject of his dissertation, completed in 1929 at the age of 23 and published the following year. This also concludes this part on first-order logic. In the next part, we continue with what made Gödel famous throughout the world, far beyond mathematics: the Incompleteness Theorem.

## Exercises

### The Completeness Theorem

**Exercise 1.** Recall Theorems A and B:

- Theorem A. If  $\mathcal{T} \models \mathcal{Q}$ , then  $\mathcal{T} \vdash \mathcal{Q}$ .
- Theorem B. If  $\mathcal{T}$  is consistent, then  $\mathcal{T}$  has a model.

Assume Theorem A. Deduce Theorem B.

**Exercise 2.** A theory  $\mathcal{T}$  (which may contain infinitely many formulas) is said to be *finitely consistent* if, for every finite subset  $\Gamma \subset \mathcal{T}$ ,  $\Gamma$  is consistent.

Show that a theory  $\mathcal{T}$  is consistent if and only if it is finitely consistent.

### Complete theory

**Exercise 3.** Let  $\mathcal{L}$  be a first-order language and let  $\mathcal{S}$  be a structure for this language. The *theory* of  $\mathcal{S}$ , denoted  $\mathcal{T}(\mathcal{S})$ , is defined as the set of all closed formulas  $\mathcal{P}$  in  $\mathcal{L}$  that are true in  $\mathcal{S}$ :

$$\mathcal{T}(\mathcal{S}) = \{\mathcal{P} \text{ closed formula} \mid \models_{\mathcal{S}} \mathcal{P}\}$$

Show that  $\mathcal{T}(\mathcal{S})$  is a complete theory.

**Exercise 4.** Let  $\mathcal{T}$  be a consistent theory. Show the equivalence between:

- (i)  $\mathcal{T}$  is complete,
- (ii) for every closed formula  $\mathcal{Q}$ , if  $\mathcal{T} \cup \{\mathcal{Q}\}$  is consistent, then  $\mathcal{T} \vdash \mathcal{Q}$ .

### Constructing a model from a complete theory

**Exercise 5.** Consider a language  $\mathcal{L}$  without an equality symbol, containing a constant 0, a unary function symbol  $s$ , and a unary predicate  $P$ . Let  $\mathcal{T}$  be a complete theory in  $\mathcal{L}$  that contains, in particular, the following two axioms (formulas):

- $\mathcal{A}_1 : P(0)$
- $\mathcal{A}_2 : \forall x (P(x) \leftrightarrow \neg P(s(x)))$

1. Construct the canonical model  $\mathcal{S}$  associated with  $\mathcal{T}$  (as in Section 3.2).
2. Construct a model  $\mathcal{S}'$  with domain  $\mathbb{N}$ , where  $s(x)$  corresponds to  $x + 1$  (the successor), and explain what  $P(x)$  represents.

**Exercise 6.** Complete the proof of Lemma 2 for the missing connectives, as well as for the quantifier  $\exists$ ; that is, for the formulas:  $\mathcal{Q} = \mathcal{P}_1 \vee \mathcal{P}_2$ ,  $\mathcal{Q} = \mathcal{P}_1 \rightarrow \mathcal{P}_2$ ,  $\mathcal{Q} = \mathcal{P}_1 \leftrightarrow \mathcal{P}_2$ ,  $\mathcal{Q} = \exists x \mathcal{P}(x)$ .

### Constructing a complete theory

**Exercise 7.** Work in the simpler setting of propositional logic. Let  $\mathcal{L}$  be a language containing only the two propositional variables  $A$  and  $B$ .

1. Explain why every formula is a closed formula. Give examples of formulas.
2. Explain why constructing a model of a theory  $\mathcal{T}$  amounts to assigning a T/F value to  $A$  and a T/F value to  $B$  that make all the formulas of  $\mathcal{T}$  true.
3. Explain why, if  $\mathcal{T}$  has a model, then  $\mathcal{T}$  is consistent.

We now want to construct (the beginning of) a complete theory starting from  $\mathcal{T}_0 = \{A\}$ . Consider the following choice for the enumeration  $(\mathcal{P}_n)_{n \geq 0}$  of the formulas of  $\mathcal{L}$ :

$$B, \quad A \vee B, \quad \neg A, \quad A \rightarrow B, \quad \dots$$

For  $n \geq 0$ , define:

- $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\mathcal{P}_n\}$  if this union is consistent,
- $\mathcal{T}_{n+1} = \mathcal{T}_n \cup \{\neg \mathcal{P}_n\}$  otherwise.

4. Explain why  $\mathcal{T}_0$  is consistent.
5. Construct  $\mathcal{T}_1, \dots, \mathcal{T}_4$ .
6. Will the complete theory  $\mathcal{T} = \bigcup_{n \geq 0} \mathcal{T}_n$  prove  $A \wedge B$ ?
7. Start again with  $\mathcal{T}'_0 = \{A\}$ , but this time consider the enumeration:

$$\neg B, \quad A \vee B, \quad \neg A, \quad A \rightarrow B, \quad \dots$$

Construct  $\mathcal{T}'_1, \dots, \mathcal{T}'_4$ .

8. Will the complete theory  $\mathcal{T}' = \bigcup_{n \geq 0} \mathcal{T}'_n$  be identical to  $\mathcal{T}$ ?

## Constructing witnesses

**Exercise 8.** Consider an initial language  $\mathcal{L}_0$  containing a single unary predicate  $P(x)$ , with no constants or functions. Let the theory  $\mathcal{T}_0$  consist of the following two axioms:

$$\mathcal{A}_1 : \exists x P(x) \quad \text{and} \quad \mathcal{A}_2 : \exists x \neg P(x)$$

1. Explain why  $\mathcal{T}_0$  is consistent (for example, by constructing a model of it).
2. Does the language  $\mathcal{L}_0$  have any closed terms? Does the theory  $\mathcal{T}_0$  have Henkin witnesses in  $\mathcal{L}_0$ ?
3. Apply Henkin's method to the two formulas  $P(x)$  and  $\neg P(x)$ . Let  $\mathcal{L}_1 = \mathcal{L}_0 \cup \{c_1, c_2\}$  be the language extended by two constants  $c_1$  and  $c_2$ . Which Henkin axioms have been added? What is the resulting theory  $\mathcal{T}_1$ ?
4. Show that  $\mathcal{T}_1$  proves  $P(c_1) \wedge (\neg P(c_2))$ . Deduce that, in every model of  $\mathcal{T}_1$ , the interpretation of  $c_1$  and the interpretation of  $c_2$  must be two distinct elements of the domain.

**Exercise 9.** Let  $\mathcal{L}'$  be an extension of a language  $\mathcal{L}$ . Let  $\mathcal{T}$  be a theory in the language  $\mathcal{L}'$  that is complete and has Henkin witnesses. Let  $\mathcal{P}(x)$  be a formula of the initial language  $\mathcal{L}$  (whose only free variable is  $x$ ).

Show that if  $\mathcal{T}$  does not prove  $\forall x \mathcal{P}(x)$ , then there is a Henkin constant  $c$  of the language  $\mathcal{L}'$  such that:

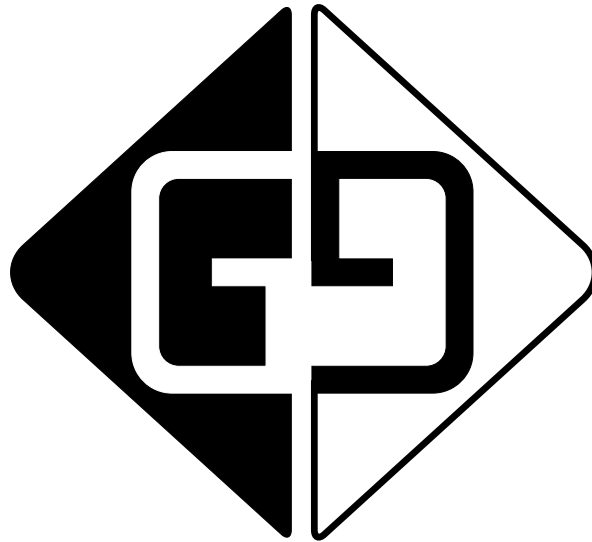
$$\mathcal{T} \vdash \neg \mathcal{P}(c).$$

What is  $c$  called in ordinary language?

---

---

## PART III



## GÖDEL'S INCOMPLETENESS THEOREMS

---

---

# An Overview of Gödel's Incompleteness Theorems

## Chapter 12

*Before embarking on the construction of the tools needed to state Gödel's theorems precisely and prove them, we give an overview of the Incompleteness Theorems and the long road ahead.*

## 1. Statements of the Theorems

### 1.1. Statement for the Arithmetic of Natural Numbers

Here is a still informal version of the First Incompleteness Theorem.

#### **Theorem 1.**

*In the theory of the natural numbers, there are true but unprovable formulas.*

We denote by  $\mathcal{PA}$  the theory of arithmetic of the natural numbers. It is a list of formulas called the *Peano axioms* (for example, “ $x + 0 = x$ ” and “ $x \times 0 = 0$ ”). They are constructed so that the natural numbers  $\mathbb{N}$  naturally satisfy these axioms.

In fact, Gödel did not speak of truth in his statement, in order to avoid discussions involving statements like the liar paradox, which asserts, “I am a liar.” Instead, he constructs a formula  $\mathcal{G}$  such that neither  $\mathcal{G}$  nor  $\neg\mathcal{G}$  is provable. The theory is then said to be *incomplete* (hence the name of the theorem).

Now, if we work in the model  $\mathbb{N}$  of the natural numbers, which satisfies the Peano axioms, then either  $\mathcal{G}$  is true or  $\neg\mathcal{G}$  is true, so one of the two formulas is indeed true but unprovable.

### 1.2. General Theorem

One might think that the Peano axioms are to blame because they are poorly chosen or too few in number. But Gödel's theorem is actually more general.

#### **Theorem 2.**

*Every consistent theory  $\mathcal{T}$  that contains  $\mathcal{PA}$  and has recursive axioms is incomplete.*

Let us forget the hypotheses for a moment and return to Peano arithmetic. Since this theory is incomplete, there is a formula  $\mathcal{P}_0$  that is not provable in  $\mathcal{PA}$ . Let  $\mathcal{T}_0 = \mathcal{PA}$ , and let  $\mathcal{T}_1 = \mathcal{T}_0 \cup \{\mathcal{P}_0\}$  be the new theory obtained by adding the unprovable formula  $\mathcal{P}_0$  to  $\mathcal{T}_0$ .

Now  $\mathcal{P}_0$  is provable in  $\mathcal{T}_1$ :  $\mathcal{T}_1 \vdash \mathcal{P}_0$ , because  $\mathcal{P}_0$  is one of the formulas in  $\mathcal{T}_1$ . Unfortunately, the general version of the Incompleteness Theorem states that this new theory  $\mathcal{T}_1$  is also incomplete: there is a formula  $\mathcal{P}_1$  such that neither  $\mathcal{P}_1$  nor  $\neg\mathcal{P}_1$  is provable. We could add  $\mathcal{P}_1$  (or  $\neg\mathcal{P}_1$ ) to construct a theory  $\mathcal{T}_2, \dots$  but this only postpones the problem indefinitely.

Thus, no matter how many unprovable formulas we add one at a time as axioms, there will always be unprovable formulas. That is incompleteness!

### 1.3. Concepts Involved

Let us review the definitions of the terms appearing in the hypotheses and the conclusion.

#### Completeness.

- A theory  $\mathcal{T}$  is *incomplete* if there is a closed formula  $Q$  such that:

$$\mathcal{T} \not\vdash Q \quad \text{and} \quad \mathcal{T} \not\vdash \neg Q$$

- If we work in a *model* of  $\mathcal{T}$ , that is, a structure in which all the formulas of  $\mathcal{T}$  are satisfied, then one of the statements  $Q$  or  $\neg Q$  is true but not provable.

#### Consistency.

- A theory  $\mathcal{T}$  is *inconsistent* if there is a closed formula  $Q$  such that:

$$\mathcal{T} \vdash Q \quad \text{and} \quad \mathcal{T} \vdash \neg Q$$

It is said to be *consistent* otherwise.

- Note carefully that the definition of “inconsistent” is not the negation of “incomplete”, because the negation of an “and” is an “or.”
- Thus, an inconsistent theory proves  $\mathcal{T} \vdash (Q \wedge \neg Q)$ , so  $\mathcal{T} \vdash \perp$ . Then, by the rule of explosion,  $\mathcal{T} \vdash Q$  (whatever the formula  $Q$  may be), so we also have  $\mathcal{T} \vdash \neg Q$ .
- An inconsistent theory is therefore of no interest, since it cannot have a model. Indeed, if it had a model, then because  $\mathcal{T} \vdash Q$  and  $\mathcal{T} \vdash \neg Q$ , the Soundness Theorem would imply that  $Q$  and  $\neg Q$  must both be true, which gives a contradiction.
- In other words, if  $\mathcal{T}$  has a model, then  $\mathcal{T}$  is consistent.

#### Containing $\mathcal{PA}$ .

- A theory is a set of formulas called axioms.
- $\mathcal{PA}$  is a specific theory whose formulas are the Peano axioms, which model the natural numbers  $\mathbb{N}$ .
- The hypothesis “ $\mathcal{T}$  contains  $\mathcal{PA}$ ” means that all the Peano axioms must be axioms of  $\mathcal{T}$ .
- The theory  $\mathcal{T}$  is then said to be “sufficiently rich,” meaning that we must be able to work with the natural numbers. This is a relatively weak requirement!

#### Recursive Axioms

- The hypothesis that  $\mathcal{T}$  has recursive axioms (the usual term is “recursively axiomatizable”) is more technical. It means that we must be able to recognize the axioms.
- More precisely, for a given formula, we must be able to decide by a computation (or an algorithm) whether or not it is an axiom of  $\mathcal{T}$ .
- Once again, this is a reasonable hypothesis: to decide which theorems can be proved, we must be able to identify the axioms used as assumptions!

## 1.4. Second Theorem

Under the same assumptions as the First Incompleteness Theorem, we have the following result.

### Theorem 3.

*A consistent theory cannot prove its own consistency.*

In concrete terms, this means that we cannot hope to verify the hypotheses of the Incompleteness Theorem while remaining within the theory  $\mathcal{T}$ ; we would have to work in a broader framework.

For example,  $\mathcal{PA}$  is generally considered to be a theory that has a model. In reality, we have implicitly accepted the existence of the natural numbers  $\mathbb{N}$ . To construct  $\mathbb{N}$  rigorously, we must work in a broader framework, for example the theory  $\mathcal{ZFC}$  (see the chapter “Peano Arithmetic”).

## 2. Consequences

### 2.1. Mathematical Consequences

Around 1900, Hilbert wanted to define a rigorous framework for the foundations of logic. During the 1800s, many mathematical concepts had already been formalized: the real numbers, limits, continuity, ... It was time to go further and completely axiomatize mathematics.

The goal was to abstract away from the physical world (continuity means drawing the graph without lifting your pencil) and write only logical formulas. However, problems quickly arose, such as Russell's paradox: it is not easy to define what a set is.

Gödel's First and Second Incompleteness Theorems shattered Hilbert's program. After Gödel, there was no longer any hope of obtaining a perfect axiomatization:

- on the one hand, if we work in a theory (sufficiently rich and consistent), there will be true but unprovable formulas,
- moreover, we cannot even prove that the theory is consistent in order to guarantee that we cannot prove both a statement and its opposite.

This was therefore fatal to Hilbert's program.

Despite this shock, Gödel's theorems had profound positive consequences extending far beyond a result concerning a small circle of logicians.

First, the tools developed by Gödel were forerunners of computer science. For example, the arithmetization of syntax (encoding every formula as a number) is a natural idea in the digital world in which we live today, but that was not the case in 1930.

More generally, Gödel was interested in determining what is computable, well before the notion of an algorithm and Turing's work (1936) on a theoretical computer, the Turing machine, and undecidability.

### The Continuum Hypothesis.

Even though Gödel constructs an explicit formula  $\mathcal{G}$  that is not provable, the only purpose of this formula is to prove the theorem.

The *continuum hypothesis* is a much more concrete statement that is also undecidable (neither  $\mathcal{CH}$  nor  $\neg\mathcal{CH}$  is provable in  $\mathcal{ZF}$ ).

The independence result was completed by Cohen in 1963, thereby resolving the first of Hilbert's 23 problems.

Let us explain this problem.

- The sets  $\mathbb{N}$ ,  $\mathbb{Z}$ , and  $\mathbb{Q}$  are countable (in bijection with  $\mathbb{N}$ ), so they have the same cardinality, even though they are infinite sets. We denote this cardinality by  $\aleph_0$  (*aleph* is the first letter of the Hebrew alphabet).

- The sets  $\mathbb{R}$ ,  $[0, 1]$ ,  $\mathbb{R}^2$ , and  $\mathbb{C}$  are uncountable; we denote their cardinality by  $2^{\aleph_0}$  (an explanation of this notation follows below).
- The continuum hypothesis states that there is no other type of infinity between  $\aleph_0$  and  $2^{\aleph_0}$ .
- In other words, if  $\aleph_1$  denotes the first type of infinity strictly greater than  $\aleph_0$ , then  $\aleph_1 = 2^{\aleph_0}$ .
- Cohen thus proves that this question cannot be answered, which means that we can choose to accept or reject  $\mathcal{CH}$  independently of everything else.

Let us return to the notation  $2^{\aleph_0}$  for the cardinality of  $\mathbb{R}$ .

For two sets  $E, F$ , we sometimes denote by  $F^E$  the set of functions  $f : E \rightarrow F$ . This is justified by the fact that if  $E$  and  $F$  are finite, the number of such functions is  $\text{Card}(F)^{\text{Card}(E)}$ . For example, if  $F = \{0, 1\}$  and  $\text{Card}(E) = n$ , then the number of functions  $f : E \rightarrow \{0, 1\}$  is  $2^n$ .

Let us explain how the real numbers in  $[0, 1]$  can be described by functions  $\mathbb{N} \rightarrow \{0, 1\}$ . For  $x \in [0, 1]$ , we compute its binary expansion:  $x = 0.100110101 \dots = \sum_{n=1}^{\infty} a_n 2^{-n}$ , where  $a_n \in \{0, 1\}$ . This is like decimal notation, but with only the digits 0 and 1. Thus, with each binary expansion, we can associate a function  $f : \mathbb{N} \rightarrow \{0, 1\}$  defined by  $f(n) = a_n$ . This correspondence is (roughly) bijective.

Thus, the cardinality of  $[0, 1]$  is that of  $\{0, 1\}^{\mathbb{N}}$ , which is  $2^{\aleph_0}$  (where  $\aleph_0$  denotes the cardinality of  $\mathbb{N}$ ).

This is also an opportunity to recall Cantor's theorem:  $\text{Card}(\mathcal{P}(E)) > \text{Card}(E)$  for every set  $E$ . It is easy to see that  $\mathcal{P}(\mathbb{N})$  corresponds to  $\{0, 1\}^{\mathbb{N}}$ . Indeed,  $A \subset \mathbb{N}$  naturally defines a real number  $x = 0.0110001 \dots$ , where there is a 1 in the  $n$ th position exactly when  $n \in A$ . Thus,  $[0, 1]$  has the same cardinality as  $\mathcal{P}(\mathbb{N})$ :  $2^{\aleph_0}$ , and this cardinality is strictly greater than  $\aleph_0$ .

## 2.2. Philosophical Consequence

The Incompleteness Theorems are among the few modern mathematical results known to nonmathematicians because of their influence on certain schools of thought.

These theorems make the fundamental difference between what is true and what is provable undeniable. This strengthens the Platonists' point of view: truth is an absolute reality, independent of the world around us.

Gödel's theorems also prove that mathematics is not merely a game of writing with abstract symbols, by showing that syntax (what can be proved) does not necessarily correspond to all of semantics (what is true).

They also raise questions about the difference between humans and machines. If we model a machine by a theory (with axioms corresponding to instructions), the machine will not be able to step outside its framework to prove its own consistency. For example, for Gödel's sentence  $\mathcal{G}$ , which says "I am not provable in  $\mathcal{PA}$ ," we must work in a broader framework, for example  $\mathcal{ZFC}$ , to be able to say that this sentence is true. Can a machine encompass the full complexity of human reasoning?

## 2.3. What the Incompleteness Theorems Are Not

### Completeness and incompleteness.

The Completeness Theorem and the Incompleteness Theorems are true theorems, proved by Gödel, and they do not contradict each other.

Recall completeness, for every formula  $\mathcal{Q}$ :

$$\mathcal{T} \models \mathcal{Q} \quad \text{iff} \quad \mathcal{T} \vdash \mathcal{Q}$$

Let us now work in the setting of a theory  $\mathcal{T} = \mathcal{PA}$  (or one containing  $\mathcal{PA}$ ) with  $\mathbb{N}$  as a model.

The Incompleteness Theorem states that there is a formula  $\mathcal{Q}$  such that:

$$\mathcal{T} \not\models \mathcal{Q} \quad \text{and} \quad \mathcal{T} \not\vdash \neg \mathcal{Q}$$

It concerns specific theories (modeling  $\mathbb{N}$ ), whereas completeness holds regardless of the structure  $\mathcal{S}$ . Incompleteness combined with completeness proves that  $\mathcal{Q}$  cannot be true in every structure (because otherwise, by completeness,  $\mathcal{Q}$  would be provable).

There are therefore structures  $\mathcal{S}$ , different from  $\mathbb{N}$ , in which  $\mathcal{Q}$  is false (so we do not have  $\mathcal{T} \models \mathcal{Q}$ ), even though  $\mathcal{S}$  satisfies the Peano axioms. These are called nonstandard models of the natural numbers.

### Are Mathematics and Logic Inconsistent?

The Second Theorem says that a theory cannot prove its own consistency. This means that, while remaining strictly within that theory, we can never be sure that it is impossible to prove both a formula  $\mathcal{Q}$  and its negation  $\neg\mathcal{Q}$ .

However, if we work in a broader framework, we can manage. For example, the consistency of arithmetic can be proved from the theory  $\mathcal{ZFC}$ . An infinite-regress problem then arises: we cannot be sure of the consistency of  $\mathcal{ZFC}$ . We would have to work in an even larger theory...

There is therefore no single universal logical framework at the foundation of mathematics. The empirical consensus in modern mathematics is to use the framework  $\mathcal{ZFC}$ , in which no one has found an inconsistency.

## 3. Road Map

Proving the First Incompleteness Theorem is a major undertaking, quite technical but with several brilliant ideas (for example, the  $\beta$  function, which encodes an arbitrary list of numbers using only two integers!). Even one hundred years later, it remains difficult to approach. One can only admire the fact that a single person accomplished all of this.

We introduce the upcoming chapters one by one.

### 3.1. Peano Arithmetic

We describe the seven Peano axioms and the induction axiom schema that govern the behavior of the natural numbers. We also describe other, weaker theories (Presburger arithmetic and Robinson arithmetic) and a stronger theory, Zermelo–Fraenkel set theory, possibly augmented by the axiom of choice,  $\mathcal{ZFC}$ .

### 3.2. Gödel Coding

This is the “arithmetization of syntax” part: every formula, and even every proof, must be encoded by a single integer. We will then be able to give an informal proof that remains in the spirit of Gödel’s First Theorem, using a diagonalization argument. We will also take the opportunity to motivate this diagonalization method through Russell’s paradox and Cantor’s diagonal argument.

### 3.3. Primitive Recursive Functions

This is one of the technical chapters. Primitive recursive functions are the functions  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  that are “easily computable.” The goal is to see that operations on formulas, for example checking that a sequence of formulas is indeed a proof, are easily computable.

We will discuss the construction of Hilbert-style proofs, which are better suited than natural-deduction proofs to algorithmic treatment.

### 3.4. Representable Functions

This is another technical chapter, making the transition between the mathematical universe of the natural numbers and the language  $\mathcal{L}_0$  of Peano arithmetic. We will prove that every primitive recursive function can be translated into a formula of  $\mathcal{L}_0$ . We thus obtain a formula that decides whether or not a proof is valid, as well as a formula expressing whether or not a statement is provable. It is in this chapter that we explain Gödel's  $\beta$  function.

### 3.5. Diagonalization

We explain how to obtain a formula  $\mathcal{G}$  that speaks about itself.

We also discuss the liar paradox and a theorem of Tarski stating that there is no formula that defines the true formulas (whereas we have seen that we can detect provable formulas).

### 3.6. Proof of the Incompleteness Theorem

We state and prove the First Incompleteness Theorem, first in the setting of Peano arithmetic and then in the setting of broader theories. The proof is based on the diagonal lemma, which allows us to obtain a sentence that says, "I am not provable."

### 3.7. The Second Incompleteness Theorem

We conclude by stating the Second Theorem and proving it without going into detail.

# Peano Arithmetic

*We study the axioms that define the usual arithmetic of the natural numbers.*

## 1. The natural numbers

### 1.1. Axiomatization

The set  $\mathbb{N} = \{0, 1, 2, \dots\}$  of natural numbers is familiar to us, and we know many of its properties:

$$x + 0 = x, \quad 0 + x = x, \quad x + y = y + x, \quad \dots$$

But can we identify the “very essence” of  $\mathbb{N}$ ? Which properties play the role of axioms, and which can be proved from these axioms? We already use this approach to define a group  $(G, \circ)$  or a vector space  $(E, +, \cdot)$  axiomatically.

We will follow this approach to obtain the axioms that define the arithmetic of  $\mathbb{N}$ . There will be seven basic axioms (for addition and multiplication) and a family of induction axioms; together, they form Peano arithmetic.

What is an axiom? From our point of view, an **axiom** is a closed formula. A set of axioms is therefore a theory. For a set of axioms  $\mathcal{A}$  and a formula  $\mathcal{Q}$ , we are often interested in whether  $\mathcal{A} \vdash \mathcal{Q}$ . We are also interested in the following equivalent assertion (by the Completeness Theorem): does  $\mathcal{A} \models \mathcal{Q}$  hold? But we will mainly work in a specific structure  $\mathcal{S}$  and discuss whether  $\models_{\mathcal{S}} \mathcal{A}$  implies  $\models_{\mathcal{S}} \mathcal{Q}$ ; that is, if the axioms are true in  $\mathcal{S}$ , is  $\mathcal{Q}$  also true in  $\mathcal{S}$ ? Thus, we will use the term *axioms* rather than *theory*, because we are not seeking to study general properties of theories. We will state and fix a set of formulas: the axioms of Peano arithmetic, denoted by  $\mathcal{PA}$ . The natural numbers  $\mathbb{N}$  form a model of  $\mathcal{PA}$ ; that is,  $\models_{\mathbb{N}} \mathcal{PA}$ .

### 1.2. Connection with the Incompleteness Theorems

Gödel’s Incompleteness Theorems apply to sufficiently “rich” theories  $\mathcal{T}$ . Concretely, the theory must contain the Peano axioms (that is,  $\mathcal{PA} \subset \mathcal{T}$ ; or be able to prove them). Requiring a theory to contain these axioms is not a major demand, since it is difficult to do mathematics without being allowed to work with the natural numbers.

The natural numbers will also appear in the following chapters because the proof of the Incompleteness Theorems relies on an *arithmetization*: every symbol, formula, and proof will be encoded by a natural number in  $\mathbb{N}$ .

### 1.3. What is $\mathbb{N}$ ?

To axiomatize  $\mathbb{N}$ , it would be useful to be able to define it. Unfortunately, this is not so straightforward.

- A first approach is to define a language  $\mathcal{L}$  containing the constant symbol 0 and a “successor” function  $S$  that plays the role of  $S(n) = n + 1$ . We then write  $1 = S(0)$ ,  $2 = S(1) = S(S(0))$ ,  $3 = S(2)$ . . .  $\mathbb{N}$  is the smallest set for which this is possible.
- We can also define  $\mathbb{N}$  using set theory: the natural numbers are constructed from the empty set  $\emptyset$ . We start with a language in which every object is a set (see the chapter “Variables and Structures”).

$$\begin{aligned} 0 &= \emptyset \\ 1 &= \{\emptyset\} = \{0\} \\ 2 &= \{\emptyset, \{\emptyset\}\} = \{0, 1\} \\ 3 &= \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} = \{0, 1, 2\} \\ &\dots \end{aligned}$$

We thus define  $S(n) = n \cup \{n\} = \{0, 1, \dots, n\}$ , denoted by  $n + 1$ .

We require 0 and the  $S(n)$  to belong to the set;  $\mathbb{N}$  is then the smallest set for which this is possible. We will return to this set-theoretic viewpoint when we discuss the axiomatic theory  $\mathcal{ZF}$ .

- $\mathbb{N}$  can also be defined uniquely in *second-order* logic if we write the induction axiom in the form:

$$\forall \mathcal{P} \quad \text{Rec}_{\mathcal{P}(x)}$$

(The induction principle  $\text{Rec}_{\mathcal{P}(x)}$  will be defined later.) Be careful: this takes us outside the framework of first-order logic because “for every” does not range over an individual variable but over a collection of formulas, which is not allowed in first-order logic.

In every case, although defining  $\mathbb{N}$  is not simple, the result does correspond to the set we know. However, there are also nonstandard models, that is, sets that satisfy the Peano axioms but are not  $\mathbb{N}$ . We could construct some by hand, but their existence will also be a consequence of the Incompleteness Theorem.

## 2. Peano axioms

The Peano axioms first consist of seven axioms defining elementary arithmetic, followed by an axiom schema for induction.

We consider the first-order language  $\mathcal{L}_0 = (0, S, +, \times)$ , where:

- 0 is a constant symbol (think of  $0 \in \mathbb{N}$ ).
- $S$  is a unary function, called the *successor* (think of  $S(x) = x + 1$ ).
- $+$  is a binary function (think of addition).
- $\times$  is a binary function (think of multiplication).
- $=$  is the equality predicate.

We write  $x \neq y$  for  $\neg(x = y)$ . The symbols  $\leq$  and  $<$  will be introduced later.

## 2.1. Elementary arithmetic

Here are the seven basic axioms:

- $\mathcal{A}_1 : \quad \forall x \quad S(x) \neq 0$
- $\mathcal{A}_2 : \quad \forall x \quad (x = 0) \vee (\exists y \quad x = S(y))$
- $\mathcal{A}_3 : \quad \forall x \quad \forall y \quad (S(x) = S(y)) \rightarrow (x = y)$
- $\mathcal{A}_4 : \quad \forall x \quad x + 0 = x$
- $\mathcal{A}_5 : \quad \forall x \quad \forall y \quad x + S(y) = S(x + y)$
- $\mathcal{A}_6 : \quad \forall x \quad x \times 0 = 0$
- $\mathcal{A}_7 : \quad \forall x \quad \forall y \quad x \times S(y) = (x \times y) + x$

Together, these axioms form *elementary arithmetic*. We will add the induction axioms immediately afterward.

Here are comments on each axiom and its interpretation in  $\mathbb{N}$ :

- $\mathcal{A}_1$ : A successor is never zero. In  $\mathbb{N}$ , we always have  $n + 1 \neq 0$ . It is important that we cannot return to the starting point, as would happen if we cycled modulo  $N$ :  $0, 1, 2, \dots, N - 1$ , then  $0$ .
- $\mathcal{A}_2$ : Every nonzero element is a successor. In  $\mathbb{N}$ , if  $n \neq 0$ , then there exists  $m \in \mathbb{N}$  such that  $n = m + 1$ . (In fact, “ $m = n - 1$ ”, but subtraction is not available.) This is the only redundant axiom, since it follows from the other axioms and induction (see the exercises). It will nevertheless be important for Robinson arithmetic  $\mathcal{Q}$  (see the end of the chapter).
- $\mathcal{A}_3$ : The successors of  $x$  and  $y$  are equal if and only if  $x$  and  $y$  are equal; in other words,  $S$  is injective. With  $\mathcal{A}_1$ , this implies that starting from  $0$ , then  $1 = S(0)$ ,  $2 = S(1)$ ,  $\dots$ , we produce only new elements. In  $\mathbb{N}$ , this means  $(n + 1 = m + 1) \rightarrow (n = m)$ . By the definition of a function, if  $x = y$ , then  $S(x) = S(y)$ .
- $\mathcal{A}_4$ : Adding  $0$  on the right changes nothing. In  $\mathbb{N}$ ,  $n + 0 = n$ . In Peano arithmetic we will prove that this is also true on the left, “ $0 + x = x$ ”, but this requires induction. We will see that this becomes problematic for Robinson arithmetic  $\mathcal{Q}$ .
- $\mathcal{A}_5$ : The successor behaves well with respect to addition. In  $\mathbb{N}$ , we would write  $n + (m + 1) = (n + m) + 1$ , a consequence of associativity.
- $\mathcal{A}_6$ : Multiplication by  $0$  gives  $0$ . In  $\mathbb{N}$ ,  $n \times 0 = 0$ . This will also be true on the left.
- $\mathcal{A}_7$ : Addition behaves well with respect to multiplication. In  $\mathbb{N}$ ,  $n \times (m + 1) = (n \times m) + n$ , a consequence of distributivity.

## 2.2. Induction axioms

We add not one axiom but infinitely many axioms, called an *axiom schema*. There is one axiom for each formula  $\mathcal{P}$  having  $x$  as a free variable.

Let  $\mathcal{P}$  be a formula of  $\mathcal{L}_0$  with a free variable  $x$ . We write  $\mathcal{P}(x)$  to indicate its dependence on  $x$ . This also allows us to write  $\mathcal{P}(y)$  instead of  $\mathcal{P}[y/x]$ .

The induction principle for the formula  $\mathcal{P}(x)$  is written:

$$\text{Rec}_{\mathcal{P}(x)} : \quad \mathcal{P}(0) \wedge [\forall x \quad \mathcal{P}(x) \rightarrow \mathcal{P}(S(x))] \quad \rightarrow \quad \forall x \quad \mathcal{P}(x)$$

In  $\mathbb{N}$ , this is the usual induction principle:

- if  $\mathcal{P}(0)$  is true (base case),
- and if  $\mathcal{P}(n) \rightarrow \mathcal{P}(n+1)$  for every  $n \in \mathbb{N}$  (inductive step),
- then  $\mathcal{P}(n)$  is true for every  $n \in \mathbb{N}$  (conclusion).

We write:

$$\mathcal{R}ec = \left\{ \mathcal{R}ec_{\mathcal{P}(x)} \mid \mathcal{P} \text{ is a formula of } \mathcal{L}_0 \text{ with a free variable } x \right\}$$

$\mathcal{R}ec$  contains infinitely many (countably many) formulas.

### Definition.

On note  $\mathcal{P}\mathcal{A} = \{\mathcal{A}_1, \dots, \mathcal{A}_7\} \cup \mathcal{R}ec$ . These are the axioms of *Peano arithmetic*.

### Remark.

To be slightly more precise in our notation, we can write:

$$\mathcal{R}ec_{\mathcal{P}(x)} : \mathcal{P}[0/x] \wedge (\forall y \mathcal{P}[y/x] \rightarrow \mathcal{P}[S(y)/x]) \rightarrow \forall x \mathcal{P}(x)$$

Furthermore, if there are other free variables  $x_1, \dots, x_n$ , we write  $\mathcal{P}(x, x_1, \dots, x_n)$  and add universal quantifiers at the beginning:

$$\mathcal{R}ec_{\mathcal{P}} : \forall x_1 \dots \forall x_n \left[ \mathcal{P}(0, x_1, \dots, x_n) \wedge [\forall x \mathcal{P}(x, x_1, \dots, x_n) \rightarrow \mathcal{P}(S(x), x_1, \dots, x_n)] \right. \\ \left. \rightarrow \forall x \mathcal{P}(x, x_1, \dots, x_n) \right]$$

This version is essential for proving  $x + y = y + x$  for every  $x, y$  by induction on  $x$ .

## 2.3. Models of $\mathcal{P}\mathcal{A}$

### Proposition 1.

$\mathbb{N}$  is a model of  $\mathcal{P}\mathcal{A}$ .

Consider the structure  $\mathcal{S}$ , which by an abuse of notation we denote by  $\mathbb{N}$ , with domain  $\mathbb{N}$  and naturally associated with  $\mathcal{L}_0$  (0 is zero,  $+$  is addition,  $\times$  is multiplication, and  $S(n) = n + 1$ ). As noted above,  $\mathbb{N}$  satisfies the axioms of  $\mathcal{P}\mathcal{A}$ ; that is,  $\models_{\mathbb{N}} \mathcal{A}_1, \dots, \models_{\mathbb{N}} \mathcal{A}_7$ , and, for every  $\mathcal{P}$ ,  $\models_{\mathbb{N}} \mathcal{R}ec_{\mathcal{P}(x)}$ . This proves the proposition.

### Corollary 1.

$\mathcal{P}\mathcal{A}$  is consistent.

Recall that this means  $\mathcal{P}\mathcal{A} \not\vdash \perp$ ; in other words, there is no formula  $\mathcal{Q}$  for which both  $\mathcal{P}\mathcal{A} \vdash \mathcal{Q}$  and  $\mathcal{P}\mathcal{A} \vdash \neg \mathcal{Q}$  hold. A theory  $\mathcal{T}$  is consistent if and only if it has a model (see the chapter “Completeness Theorem”). Since  $\mathbb{N}$  is a model of  $\mathcal{P}\mathcal{A}$ ,  $\mathcal{P}\mathcal{A}$  is consistent.

However, there are other models of  $\mathcal{P}\mathcal{A}$ , called *nonstandard* models. They are all infinite and contain  $\mathbb{N}$ . This is a very interesting point: we have succeeded in formalizing the essential properties of  $\mathbb{N}$ , yet this is not enough to characterize  $\mathbb{N}$  uniquely. These seemingly paradoxical ideas lie at the heart of this book. In fact, no first-order axiomatization can characterize  $\mathbb{N}$  uniquely.

**Warning.** Gödel’s First Incompleteness Theorem begins with the hypothesis “If  $\mathcal{P}\mathcal{A}$  is consistent, ...”. Why is this hypothesis needed, since we have just said that it is true? Working only in the language  $\mathcal{L}_0$  of Peano arithmetic, we cannot prove within that language that  $\mathcal{P}\mathcal{A}$  is consistent. This is precisely Gödel’s Second Incompleteness Theorem, which says that no sufficiently rich theory can prove its own consistency.

Here, we assume that  $\mathbb{N}$  exists. Thus  $\mathcal{PA}$  has a model and is necessarily consistent. This certainty rests on a metamathematical acceptance of the existence of  $\mathbb{N}$ ; that is, we must work in a more general theory (for example,  $\mathcal{ZFC}$ ) to confirm this construction.

### 3. Properties of Peano arithmetic

We deliberately chose a limited number of axioms. We must now recover the usual arithmetic properties from them. We will prove very few, since that would belong more to a mathematics course than to a logic course. Above all, this is an ideal opportunity to use formal proof software.

#### 3.1. Using the basic axioms

**Lemma 1.**

$$2 + 2 = 4$$

Yes, nothing can be taken for granted! We hardly dare call this a “theorem”, yet we must justify that  $\mathcal{PA} \vdash “2 + 2 = 4”$ .

**Remark.** We write  $1 = S(0)$ ,  $2 = S(1)$ ,  $3 = S(2)$ ,  $4 = S(3)$ . These are only notational conventions:  $1, 2, 3, 4$  are not symbols of the language but syntactic abbreviations. Without them, we would have to write  $S(S(0)) + S(S(0)) = S(S(S(S(0))))$ .

*Proof.*

$$\begin{aligned} 2 + 2 &= 2 + S(1) && \text{by the definition of } 2 = S(1) \\ &= S(2 + 1) && \text{by } \mathcal{A}_5 \\ &= S(2 + S(0)) && \text{by the definition of } 1 = S(0) \\ &= S(S(2 + 0)) && \text{by } \mathcal{A}_5 \\ &= S(S(2)) && \text{by } \mathcal{A}_4 \\ &= S(3) && \text{by the definition of } S(2) = 3 \\ &= 4 && \text{by the definition of } S(3) = 4 \end{aligned}$$

□

Without induction, no general algebraic property can be proved.

#### 3.2. Using induction

**Lemma 2.**

$$\forall x \quad 0 + x = x$$

This resembles axiom  $\mathcal{A}_4$ , “ $\forall x \quad x + 0 = x$ ”, but we do not yet know that addition is commutative.

*Proof.* Let  $\mathcal{P}(x) : 0 + x = x$ .

We write the induction proof in the usual mathematical style:

- Base case.  $\mathcal{P}(0)$ :  $0 + 0 = 0$ . This is true by  $\mathcal{A}_4$  (it is the 0 on the right that applies, not the one on the left!).

- Inductive step. Fix  $n$ . Assume  $\mathcal{P}(n)$  is true and prove  $\mathcal{P}(S(n))$  (the equivalent of  $\mathcal{P}(n+1)$ ).

$\mathcal{P}(n)$  is true  
 therefore  $0 + n = n$   
 therefore  $S(0 + n) = S(n)$       because  $S$  is a function  
 therefore  $0 + S(n) = S(n)$       by axiom  $\mathcal{A}_5$   
 therefore  $\mathcal{P}(S(n))$  is true.

Thus  $\mathcal{P}(n) \rightarrow \mathcal{P}(S(n))$ , and in the logical notation of  $\mathcal{PA}$ , we have proved  $\mathcal{P}(x) \rightarrow \mathcal{P}(S(x))$  for every  $x$ .

- Conclusion. By induction, here  $\text{Rec}_{\mathcal{P}(x)}$ , we deduce “ $\forall x \mathcal{P}(x)$ ”, that is, “ $\forall x 0 + x = x$ ”.

□

**Lemma 3.**

$$\forall x \forall y \quad S(x) + y = S(x + y)$$

This is like axiom  $\mathcal{A}_5$ , but with the roles reversed.

*Proof.* Fix  $x$ . We proceed by induction on  $y$ . Let  $\mathcal{P}(y) : S(x) + y = S(x + y)$ .

- Base case.  $\mathcal{P}(0)$ :  $S(x) + 0 = S(x) = S(x + 0)$ . This holds.
- Inductive step. Assume  $\mathcal{P}(y)$ .

$S(x) + y = S(x + y)$       by the induction hypothesis  
 therefore  $S(S(x) + y) = S(S(x + y))$       because  $S$  is a function  
 therefore  $S(x) + S(y) = S(x + S(y))$       by applying  $\mathcal{A}_5$  on the left and right  
 therefore  $\mathcal{P}(S(y))$  is true.

Thus, for every  $y$ ,  $\mathcal{P}(y) \rightarrow \mathcal{P}(S(y))$ .

- Conclusion. By induction,  $\forall y \mathcal{P}(y)$ . Since this holds for every  $x$ , we have  $\forall x \forall y \quad S(x) + y = S(x + y)$ .

Remark. To be rigorous, we should write  $\mathcal{P}(x, y) : S(x) + y = S(x + y)$ , then prove  $\forall x \mathcal{P}(x, 0)$  and  $\forall x \forall y \mathcal{P}(x, y) \rightarrow \mathcal{P}(x, S(y))$ , which leads to  $\forall x \forall y \mathcal{P}(x, y)$ . □

**Lemma 4.**

$$\forall x \forall y \quad x + y = y + x$$

This is the expected commutativity of addition.

*Proof.* Fix  $x$ . We proceed by induction on  $y$ . Let  $\mathcal{P}(y) : x + y = y + x$ .

- Base case.  $\mathcal{P}(0)$ :  $x + 0 = x$  (by axiom  $\mathcal{A}_4$ ) and  $0 + x = x$  (by Lemma 2). Therefore  $x + 0 = 0 + x$ .
- Inductive step. Assume  $\mathcal{P}(y)$  is true.

$x + y = y + x$   
 therefore  $S(x + y) = S(y + x)$   
 therefore  $x + S(y) = S(y) + x$       by  $\mathcal{A}_5$  on the left and Lemma 3 on the right  
 therefore  $\mathcal{P}(S(y))$  is true.

- Conclusion. For every  $x$ :  $\forall y \quad x + y = y + x$ .

□

### 3.3. Other operations

- **Inequality.**  $x \neq y$  means  $\neg(x = y)$ .
- **Nonstrict inequality.**  $x \leq y$  means  $\exists z \ x + z = y$  (recall that we are modeling the natural numbers; think of  $z$  as an element of  $\mathbb{N}$ ).
- **Strict inequality.**  $x < y$  means  $(x \leq y) \wedge (x \neq y)$ .
- **Subtraction.** Be careful:  $x - y$  is not a function because it is not always well-defined; for example,  $2 - 3$  has no meaning here. For  $x \leq y$ , however, there exists  $z$  such that  $x + z = y$ , and we could define  $y - x = z$ .
- Nor is there a well-defined division  $x/y$ .

There is now plenty of work to do, for example, proving:

$$\begin{aligned} (x \leq y) \wedge (y \leq z) &\rightarrow x \leq z \\ x \leq y &\rightarrow x \times z \leq y \times z \\ &\dots \end{aligned}$$

### 3.4. Definable sets

Fix a first-order language  $\mathcal{L}$  and an associated structure  $\mathcal{S}$  with domain  $E$ .

**Definition.**

A set  $A \subset E$  is **definable** if there is a formula  $\mathcal{P}(x)$  of  $\mathcal{L}$  such that:

$$A = \{a \in E \mid \models_{\mathcal{S}} \mathcal{P}(a)\}$$

In other words,  $A$  is exactly the set of all  $a$  for which  $\mathcal{P}(a)$  is true.

**Example.**

- The set of even numbers in  $\mathbb{N}$  is definable by the formula:

$$\mathcal{P}(x): \exists y \ x = y + y$$

- In  $\mathbb{N}$ , the set of prime numbers is definable by the formula:

$$\mathcal{P}(x): (x \neq 1) \wedge \left[ \forall y \ \forall z \ (x = y \times z) \rightarrow (y = 1) \vee (z = 1) \right]$$

When working formally in Peano arithmetic (independently of the model  $\mathbb{N}$ ), this formula serves as the definition of a prime number (with the abbreviation  $1 = S(0)$ ).

### 3.5. Strong induction

In an ordinary induction proof, we assume  $\mathcal{P}(n)$  to deduce  $\mathcal{P}(n+1)$ . In strong induction, we assume that  $\mathcal{P}(0), \mathcal{P}(1), \dots, \mathcal{P}(n)$  are true to deduce that  $\mathcal{P}(n+1)$  is true. This is sometimes very useful.

Here is the strong induction formula for  $\mathcal{P}$ :

$$\text{RecF}_{\mathcal{P}(x)}: \left[ \forall x \ (\forall y \ y < x \rightarrow \mathcal{P}(y)) \rightarrow \mathcal{P}(x) \right] \rightarrow \forall x \ \mathcal{P}(x)$$

Over  $\mathbb{N}$ , this expresses that the strong inductive step  $[\mathcal{P}(0) \wedge \dots \wedge \mathcal{P}(n-1)] \rightarrow \mathcal{P}(n)$  entails  $\forall n \ \mathcal{P}(n)$ .

The base case is implicit. Taking  $x = 0$  in the brackets, the left-hand side of the implication, “ $\forall y \quad y < 0 \rightarrow \mathcal{P}(y)$ ”, is always true (no  $y$  is strictly less than 0 in  $\mathbb{N}$ ); therefore  $\mathcal{P}(0)$  must be true for the implication to be true.

As an exercise, one could show that strong induction is equivalent to ordinary induction.

## 4. Other arithmetics

### 4.1. Presburger arithmetic

This arithmetic is simpler than Peano arithmetic because it has no multiplication. The language is  $\mathcal{L} = (0, S, +)$ , and we consider only axioms  $\mathcal{A}_1, \dots, \mathcal{A}_5$  and  $\mathcal{R}_{ec}$ . We denote this theory by  $\mathcal{PR}$ . Although multiplication is absent, we can define  $3x$  as  $x + x + x$ ; in contrast, a formula “ $x \times y$ ” is not allowed.

#### Theorem 1.

*$\mathcal{PR}$  is consistent and complete.*

Recall what this means. For any (closed) formula  $\mathcal{Q}$ :

- $\mathcal{PR}$  is consistent, so it cannot prove both  $\mathcal{Q}$  and  $\neg\mathcal{Q}$ .
- $\mathcal{PR}$  is complete; consequently, it proves  $\mathcal{Q}$  or  $\neg\mathcal{Q}$ .

The proof is rather technical and uses “quantifier elimination”: a formula of  $\mathcal{L}$  is transformed into an equivalent quantifier-free formula by introducing congruence modulo integers  $n$ . We first transform  $\forall$  into  $\exists$ , since “ $\forall x \quad \mathcal{P}(x)$ ” is equivalent to “ $\neg(\exists x \quad \neg\mathcal{P}(x))$ ”. For example, “ $y$  is even” is:

$$\exists x \quad y = x + x$$

that is,  $\exists x \quad y = 2x$ , which is equivalent to:

$$y \equiv 0 \pmod{2}$$

Thus, using congruence, the formula has no quantifiers. In general, a closed formula yields an algebraic expression modulo an integer  $n$ . The formula is provable if and only if this expression is satisfied. Thus, either a formula or its negation is provable.

### 4.2. Robinson arithmetic $\mathcal{Q}$

We define a new axiomatic theory, with addition and multiplication but without induction. We denote by  $\mathcal{Q}$  the theory consisting of axioms  $\mathcal{A}_1, \dots, \mathcal{A}_7$ , without induction axioms. This time, axiom  $\mathcal{A}_2$ , which states that every nonzero element is a successor, is essential.

#### Theorem 2.

*$\mathcal{Q}$  is consistent but incomplete.*

The theory  $\mathcal{Q}$  is consistent because  $\mathbb{N}$  is a model of it.

To prove that it is incomplete, we show that the formula

$$\mathcal{G} : \quad \forall x \quad 0 + x = x$$

is not provable, and neither is  $\neg\mathcal{G}$ . (The letter  $G$  stands for “Gödel”.)

Do not confuse  $\mathcal{G}$  with axiom  $\mathcal{A}_4$ , “ $\forall x \quad x + 0 = x$ ”. Recall that proving  $\mathcal{G}$  in Peano arithmetic required induction, which is not allowed here.

*Proof.*

- $\neg\mathcal{G}$  is not provable.

Suppose for a contradiction that  $\neg\mathcal{G}$  is provable. Then  $\mathcal{Q} \vdash \neg\mathcal{G}$  and, by the Soundness Theorem,  $\mathcal{Q} \models \neg\mathcal{G}$ . But  $\models_{\mathbb{N}} \mathcal{Q}$ , so  $\models_{\mathbb{N}} \neg\mathcal{G}$ . Yet  $\mathcal{G}$  is true in  $\mathbb{N}$ , a contradiction.

- $\mathcal{G}$  is not provable.

We follow the same idea: find a model of  $\mathcal{Q}$  in which  $\mathcal{G}$  is false. It cannot be  $\mathbb{N}$ , where  $\mathcal{G}$  is true.

Let  $\mathcal{S}$  have domain  $\mathbb{N} \cup \{a, b\}$ . The objects  $a$  and  $b$  are new and distinct; think of them as two infinities  $\infty_a$  and  $\infty_b$ .

For two natural numbers, addition is the usual addition, and we set:

- $a + 0 = a$
- $0 + a = b$

We also have  $S(a) = a$  and  $S(b) = b$  (in the spirit of “ $\infty + 1 = \infty$ ”); we do not need the details of the other operations below.

One can check that all axioms  $\mathcal{A}_1, \dots, \mathcal{A}_7$  are satisfied (see the exercises). On the other hand,  $\mathcal{G}$  is false because  $0 + a = b$  and  $a \neq b$ .

As before, if  $\mathcal{Q} \vdash \mathcal{G}$ , then  $\mathcal{Q} \models \mathcal{G}$ , hence  $\models_{\mathcal{S}} \mathcal{G}$ , contradicting our construction.

□

But is this the end of our story? The goal of Gödel and of this book is an Incompleteness Theorem, and we have just found an incomplete theory. Why continue? If we are dissatisfied because  $\mathcal{Q}$  proves neither  $\mathcal{G}$  nor  $\neg\mathcal{G}$ , we could extend it to  $\mathcal{Q}_1 = \mathcal{Q} \cup \{\mathcal{G}\}$ , which proves  $\mathcal{G}$ . But  $\mathcal{Q}_1$  is not complete either. There exists  $\mathcal{G}_1$  such that neither  $\mathcal{Q}_1 \vdash \mathcal{G}_1$  nor  $\mathcal{Q}_1 \vdash \neg\mathcal{G}_1$  (see the exercises). We could then consider  $\mathcal{Q}_2 = \mathcal{Q}_1 \cup \{\mathcal{G}_1\} \dots$ , merely postponing the problem.

We need a general statement showing that we will never obtain a complete theory. This is the power of Gödel’s Incompleteness Theorem: whatever the theory, it will be incomplete, so adding an unprovable formula will not produce a complete theory.

### 4.3. The axiomatic theories $\mathcal{ZF}$ and $\mathcal{ZFC}$

Zermelo–Fraenkel set theory, denoted by  $\mathcal{ZF}$ , is an axiomatic set theory that provides, among other things, foundations for arithmetic. The language is  $\mathcal{L} = \{\in\}$  (with equality), and every object is a set. See “Variables and Structures” for an introduction. Thus, “ $x \in y$ ” means “the set  $x$  belongs to the set  $y$ ”. For example,  $\{0, 1\} \in \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$ .

It is a more modern and much more powerful theory than Peano arithmetic. However, we will not need it for the Incompleteness Theorems, so this brief introduction to  $\mathcal{ZF}$  will not be used later.

We have already seen how to construct the natural numbers from the empty set:

$$0 = \emptyset, \quad 1 = \{\emptyset\}, \quad 2 = \{\emptyset, \{\emptyset\}\} = \{0, 1\}, \quad 3 = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\} = \{0, 1, 2\}, \quad \dots$$

and, more generally,  $S(n) = n \cup \{n\}$ .

$\mathcal{ZF}$  is governed by axioms for manipulating sets. We list them without all the details:

- **Empty set axiom:** there exists a set containing no elements.
- **Axiom of infinity:** it guarantees the existence of an infinite set (containing all the natural numbers).
- **Axiom of extensionality:** two sets having the same elements are equal.

$$\forall x \forall y \quad (\forall z (z \in x \leftrightarrow z \in y) \rightarrow x = y)$$

- **Axiom of foundation:** every nonempty set  $x$  has an element  $y$  disjoint from  $x$ . This rules out membership cycles (such as  $x \in x$ ).

$$\forall x \neq \emptyset \quad \exists y \in x \quad (y \cap x = \emptyset)$$

- **Axiom of union:** for a set  $x$ , there exists a set  $\bigcup_x$  that is the union of all sets belonging to  $x$ .

$$\forall x \exists y \forall z (z \in y \leftrightarrow \exists t (t \in x) \wedge (z \in t))$$

For example, if  $x = \{\{1, 2\}, \{2, 3\}, \{4, 5\}\}$ , then  $\bigcup_x = \{1, 2, 3, 4, 5\}$  (1, 2, ... are regarded as sets).

- **Axiom of pairing:** given two sets  $x$  and  $y$ , there exists a set  $\{x, y\}$  consisting exactly of  $x$  and  $y$ .  
Example: if  $x = \{\{1, 2\}, \{3\}\}$  and  $y = \{3, 4\}$ , then  $\{x, y\} = \{\{\{1, 2\}, \{3\}\}, \{3, 4\}\}$ . (This is not the union of  $x$  and  $y$ .)
- **Power set axiom:** for a set  $x$ , there exists a set  $\mathcal{P}(x)$  containing all subsets of  $x$ . For example, if  $x = \{0, 1\}$ , then  $\mathcal{P}(x) = \{\emptyset, \{0\}, \{1\}, \{0, 1\}\}$ .
- **Axiom schema of separation:** it allows subsets to be defined using a formula  $\mathcal{P}$ ; thus,  $\{x \in y \mid \mathcal{P}(x)\}$  is a set.
- **Axiom schema of replacement:** it allows the elements of a set to be replaced by other objects; the result remains a set.

These axioms form  $\mathcal{ZF}$ .

The following axiom is often added:

- **Axiom of choice  $\mathcal{AC}$ :** for every set  $A$ , there exists a function assigning to every nonempty subset of  $A$  an element of that subset.

This axiom is independent of  $\mathcal{ZF}$  (neither it nor its negation can be proved from  $\mathcal{ZF}$ ), so it may be added or omitted as needed for infinite sets. It occurs in different forms (Zorn's lemma, Krull's theorem), is essential for certain classical proofs (existence of bases in infinite-dimensional vector spaces, existence of non-Lebesgue-measurable subsets of  $\mathbb{R}$ ), and can simplify others (the Hahn–Banach theorem in the countable case). If we adopt the axiom of choice, the resulting theory is denoted by  $\mathcal{ZFC}$ . This is the standard framework for modern mathematics.

# Gödel Coding

*We turn symbols, formulas, and even proofs into natural numbers. We will then outline the proof of the First Incompleteness Theorem.*

## 1. Coding symbols, formulas, and proofs

### 1.1. Motivation

The goal is to encode each formula as a natural number so that it can be processed mathematically and algorithmically. A useful analogy is ASCII, which encodes each character as a natural number ('A': 65, 'B': 66, ...). To encode a text (a string of characters), we simply concatenate the natural numbers ('BAC': 66/65/67). Since a computer does not encode natural numbers using the digits 0 through 9 but only the symbols 0 and 1, our ASCII characters are written using 8 bits:

A : 01000001  
B : 01000010  
Z : 01011010

For a string of characters, we simply concatenate the code of each character into a single natural number.

'BAC' : 010000100100000101000011

Decoding is easy: first divide the code into blocks of 8 bits.

There is an important distinction between a number and a representation of that number:

7, VII, seven,  $111_2$ , SSSSSS(0)

are different representations of the same natural number denoted by '7' (here in Arabic numerals, Roman numerals, words, binary notation, and the language  $\mathcal{L}_0$  of Peano arithmetic). This is a familiar distinction in computer science. There is a deep philosophical dimension to naming things, even though the name does not change the thing itself. Just as the word *table* does not have four legs, arithmetic symbols serve only as a formal medium for representing natural numbers.

What is truly fundamental below is the existence of an encoding and a decoding procedure. Their inner workings are not so important and vary from one book to another. In any case, we will see that this coding is utterly unusable in practice.

### 1.2. Coding symbols

Consider the first-order language  $\mathcal{L}_0 = (0, S, +, \times)$  for Peano arithmetic. We must encode the symbols of the language, the connectives, the quantifiers, and the variables  $(x_i)_{i \geq 1}$ .

We choose (arbitrarily) the following coding (see Peter Smith's book):

|        |          |        |               |                   |           |           |     |     |     |     |     |     |          |
|--------|----------|--------|---------------|-------------------|-----------|-----------|-----|-----|-----|-----|-----|-----|----------|
| $\neg$ | $\wedge$ | $\vee$ | $\rightarrow$ | $\leftrightarrow$ | $\forall$ | $\exists$ | $=$ | $($ | $)$ | $0$ | $S$ | $+$ | $\times$ |
| 1      | 3        | 5      | 7             | 9                 | 11        | 13        | 15  | 17  | 19  | 21  | 23  | 25  | 27       |

|         |         |         |       |         |       |         |
|---------|---------|---------|-------|---------|-------|---------|
| $x_1/x$ | $x_2/y$ | $x_3/z$ | $x_4$ | $\dots$ | $x_i$ | $\dots$ |
| 2       | 4       | 6       | 8     | $\dots$ | $2i$  | $\dots$ |

For example, the code of the symbol “ $\exists$ ” is  $c = 13$ . The symbol with code  $c = 21$  is “0” (the zero constant of the language). The variable  $x_i$  is encoded by the natural number  $2i$ .

To decode a code, simply read the table from bottom to top:

- if  $c$  is odd, find the symbol to which it corresponds;
- if  $c$  is even, the symbol is the variable  $x_{c/2}$ .

Remarks:

- For some proofs, we can limit the number of connectives, for example to  $(\neg, \rightarrow, \forall)$  or to  $(\neg, \vee, \forall)$ .
- For decoding, it is important that no code be equal to 0.

### 1.3. Coding a formula

Consider a word in the language  $\mathcal{L}_0$ , that is, a finite sequence of symbols. It may be a formula “ $\forall S(x) = y$ ” or a meaningless sequence of symbols “ $S(\forall) \rightarrow \neg \times 0$ ”.

Let  $c_1, c_2, c_3, \dots$  be the codes of the symbols in this word (read from left to right). These codes will be the exponents in a product of prime numbers:

$$2^{c_1} \times 3^{c_2} \times 5^{c_3} \times \dots$$

Let  $\pi_1 = 2, \pi_2 = 3, \pi_3 = 5, \dots$  be the ordered sequence of prime numbers. We can now formalize our construction:

- Start with a word (or formula)  $\mathcal{P}$ , a sequence of symbols  $a_1, a_2, \dots, a_l$  of  $\mathcal{L}_0$ .
- Associate its code  $c_i$  with each symbol  $a_i$ .
- The **Gödel number**  $n(\mathcal{P})$  of the formula  $\mathcal{P}$  is:

$$n = \prod_{i=1}^l \pi_i^{c_i}$$

Remarks. This is generally a very large number. The construction also applies to any word in  $\mathcal{L}_0$ , even if it is not a formula.

#### Example.

- $\mathcal{P} : “\neg(x = 0)”$ 
  - $[a_1, \dots, a_6] = [\neg, (, x, =, 0, )]$
  - $[c_1, \dots, c_6] = [1, 17, 2, 15, 21, 19]$
  - $n(\mathcal{P}) = 2^1 \cdot 3^{17} \cdot 5^2 \cdot 7^{15} \cdot 11^{21} \cdot 13^{19}$
- “2” does not exist in  $\mathcal{L}_0$ ; its representation in  $\mathcal{L}_0$  is “SS0”, whose Gödel number is  $2^{23} \cdot 3^{23} \cdot 5^{21}$ .

## 1.4. Decoding

Decoding is based on prime factorization (also called the fundamental theorem of arithmetic), and in particular on the uniqueness of this factorization.

### Theorem 1.

For every natural number  $n \geq 1$ , there exist a natural number  $l \geq 1$  and natural numbers  $\alpha_1, \dots, \alpha_l \geq 0$  such that:

$$n = \pi_1^{\alpha_1} \times \pi_2^{\alpha_2} \times \dots \times \pi_l^{\alpha_l}$$

Moreover, the  $\alpha_i$  are unique.

Thus, to decode a Gödel number  $n$ , that is, recover the encoded formula, we first factor  $n$  into primes. The exponents  $c_1, \dots, c_l$  are the codes that recover the symbols of the formula one by one.

There is only one way to decode a number. Mathematically, this means that the coding is injective:

$$\text{If } n(\mathcal{P}) = n(\mathcal{Q}) \text{ then } \mathcal{P} = \mathcal{Q}.$$

### Example.

$$n = 35\,872\,267\,500$$

- Prime factorization:  $n = 2^2 \times 3^{15} \times 5^4$ .
- The exponents give the codes  $[2, 15, 4]$ .
- Each code decodes into a symbol:  $[x, =, y]$ .
- Thus, the formula encoded by  $n$  is “ $x = y$ ”.

## 1.5. Coding proofs

We also want to encode an entire proof as a single natural number. Here, a proof is a finite sequence of formulas  $(\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_L)$ . How do we encode a proof? Each  $\mathcal{P}_j$  is encoded by its Gödel number  $n_j = n(\mathcal{P}_j)$ . We then associate with the proof its **super-Gödel number**:

$$N = 2^{n_1} \cdot 3^{n_2} \dots \pi_L^{n_L}$$

that is:

$$N = \prod_{j=1}^L \pi_j^{n(\mathcal{P}_j)}$$

Remarks:

- This is a very, very large number. We will never compute it in practice.
- We will never mix Gödel numbers and super-Gödel numbers. They are two distinct encodings. We will know beforehand whether a given natural number denotes a Gödel number or a super-Gödel number.

How do we decode a super-Gödel number?

- Factor  $N$  into primes.
- Extract the list of exponents  $(n_1, \dots, n_L)$ .
- Decode each Gödel number  $n_j$  into a formula  $\mathcal{P}_j$ .
- This yields the list of formulas  $(\mathcal{P}_1, \dots, \mathcal{P}_L)$ .

## 2. Decoding algorithm

We have managed to encode symbols, formulas, and proofs as natural numbers. We will have to show that the corresponding operations on natural numbers are “sufficiently simple”, which will correspond to the notion of a *primitive recursive function* discussed in the following chapters. For now, we approach this issue algorithmically to prepare for the work ahead.

We begin with the encoding and decoding operations. For these operations to be “sufficiently simple” means that they can be programmed using only bounded “for” loops (the number of iterations is bounded as a function of the input). This means that we do not allow “while” loops for which we do not know in advance when the stopping condition will be reached.

### 2.1. Decoding a symbol code

It is easy to write an algorithm `decode_symbol_code(c)` that takes a natural number  $c$  as input (a valid symbol code) and returns the corresponding symbol  $a \in \mathcal{L}_0$  (see the previous section). It does not even require a “for” loop.

**Algorithm** (`decode_symbol_code(c)`).

- – Input: a natural number  $c$ .
- – Output: a symbol  $a$  of  $\mathcal{L}_0$ .
- If  $c = 1$ ,  $a \leftarrow “\neg”$ .
- If  $c = 3$ ,  $a \leftarrow “\wedge”$ .
- ...
- If  $c = 27$ ,  $a \leftarrow “\times”$ .
- If  $c$  is even,  $a \leftarrow x_{c/2}$ .
- Return  $a$ .

### 2.2. Decoding a Gödel number

**Lemma 1.**

Let  $n = \pi_1^{\alpha_1} \times \cdots \times \pi_l^{\alpha_l}$  with  $\alpha_i \geq 1$  ( $i = 1, \dots, l$ ).

- $l \leq \log_2(n) \leq n$
- $\alpha_i \leq \log_2(n) \leq n$  ( $i = 1, \dots, l$ )

*Proof.* Recall that, by definition,  $\log_2(2^n) = n$ . Since  $2^n \geq n$ , we have  $n \geq \log_2(n)$ .

- $n = \pi_1^{\alpha_1} \cdots \pi_l^{\alpha_l} \geq \pi_1 \times \cdots \times \pi_l \geq 2^l$ , so  $\log_2(n) \geq l$ .
- $n = \pi_1^{\alpha_1} \cdots \pi_l^{\alpha_l} \geq \pi_i^{\alpha_i} \geq 2^{\alpha_i}$ , so  $\log_2(n) \geq \alpha_i$ .

□

The loop in the following algorithm is bounded by  $K = \lfloor \log_2(n) \rfloor$ . A meticulous reader who considers computing  $\log_2(n)$  complicated may replace  $\log_2(n)$  with  $n$ .

**Algorithm** (decode\_godel\_number( $n$ )).

- – Input: a Gödel number  $n$ .
- Output: a word/formula  $[a_1, \dots, a_l]$ .
- $list \leftarrow [ ]$  (empty list).
- For  $k$  from 1 to  $K = \lfloor \log_2(n) \rfloor$ :
  - $c \leftarrow \text{exponent}(n, \pi_k)$
  - If  $c > 0$ , append  $\text{decode\_symbol\_code}(c)$  to  $list$ .
- Return  $list$ .

We leave it to the reader to write the algorithm for decoding a super-Gödel number  $N$ .

## 2.3. Prime numbers

We allow the basic arithmetic operations  $+$ ,  $-$ ,  $\times$ ,  $/$ . However, we must ensure that operations involving prime numbers are easy to perform.

First, the divisibility test  $\text{is\_divisible}(n, d)$ , which decides whether  $d$  divides  $n$ , would be easy to program with a “while” loop. Here is the algorithm using a “for” loop.

**Algorithm** (is\_divisible( $n, d$ )).

- – Input: two natural numbers  $n, d$ .
- Output: True iff  $d$  divides  $n$ .
- For  $i$  from 1 to  $n$ :
  - $n \leftarrow n - d$
  - If  $n = 0$ , return True (and end the loop).
  - If  $n < 0$ , return False (and end the loop).

From this we obtain an algorithm  $\text{is\_prime}(n)$  that returns True when  $n$  is prime and False otherwise, by testing whether there is a divisor  $d$  among  $2, 3, \dots, n-1$ .

Next, we can recover the exponent of  $p$  in the prime factorization of a natural number  $n$ . For example,  $\text{exponent}(2^a 3^b 5^c, 3)$  is  $b$ .

**Algorithm** (exponent( $n, p$ )).

- – Input: a natural number  $n$  and a prime number  $p$ .
- Output: the exponent  $e$  of  $p$  in  $n$ .
- $e \leftarrow 0$ .
- For  $k$  from 1 to  $K = \lfloor \log_2(n) \rfloor$ :
  - If  $\text{is\_divisible}(n, p)$ :
    - $e \leftarrow e + 1$
    - $n \leftarrow n/p$
  - Otherwise, exit the loop.
- Return  $e$ .

Finally, we have used throughout the list  $\pi_1 = 2, \pi_2 = 3, \dots$  of all prime numbers. We must verify that we can compute this list as far as needed. We program a function `next_prime(n)` that returns the first prime greater than  $n$ . Bertrand's postulate (proved by Chebyshev) states that there is always a prime between  $n$  and  $2n$ . It is therefore enough to test  $n + 1, \dots, 2n$  for primality. We then easily obtain a function `prime_list(K)` that returns the list of all primes less than  $K$ .

### 3. Operations on formulas

We now justify that, given a Gödel number  $n$ , we can “easily” determine whether the corresponding formula has a given property. For example, is the formula well-formed? Is it a closed formula? Does it consist of a single variable? And for a super-Gödel number  $N$ : does it encode a correct proof?

#### 3.1. Formulas

- `is_variable(n)` tests whether the formula with Gödel number  $n$  is one of the variables  $x_i$ .
- `is_term(n)` tests whether the expression with Gödel number  $n$  is a term of  $\mathcal{L}_0$ . Recall that the terms are the constant 0, the variables  $x_i$ , and, if  $t_1$  and  $t_2$  are terms, then  $S(t_1)$ ,  $t_1 + t_2$ , and  $t_1 \times t_2$  are also terms.
- `is_formula(n)` returns True when  $n$  encodes a valid formula of  $\mathcal{L}_0$  (for example, “ $x = \forall$ ” is not valid).
- `is_closed_formula(n)` returns True if  $n$  encodes a formula with no free variables.

Here, for example, is the algorithm for testing whether  $n$  encodes a term of  $\mathcal{L}_0$ .

**Algorithm** `is_term(n)`:

- – Input: a natural number  $n$ .
- – Output: True iff  $n$  encodes a term of  $\mathcal{L}_0$ .
- $\mathcal{P} \leftarrow \text{decode\_godel\_number}(n)$
- If  $\mathcal{P} = “0”$ , return True.
- If  $\mathcal{P}$  is a variable, return True.
- If  $\mathcal{P}$  has the form “ $S(t)$ ”, return `is_term(t)`.
- If  $\mathcal{P}$  has the form “ $t_1 + t_2$ ”, return `is_term(t_1) AND is_term(t_2)`.
- If  $\mathcal{P}$  has the form “ $t_1 \times t_2$ ”, return `is_term(t_1) AND is_term(t_2)`.
- Otherwise, return False.

This is a recursive function whose depth is less than the length of  $\mathcal{P}$ , and hence less than  $\log_2(n)$ . We could write an algorithm for it using loops and stacks.

### 3.2. Proof

It is essential that we be able to decide efficiently whether a given proof is valid. This is the role of the function  $\text{is\_proof}(N, n)$ .

- Let  $N$  be a super-Gödel number representing a sequence of formulas  $\mathcal{P}_1, \dots, \mathcal{P}_L$ .
- Let  $n$  be a Gödel number representing a formula  $\mathcal{Q}$ .
- The function  $\text{is\_proof}(N, n)$  returns True iff  $\mathcal{P}_L = \mathcal{Q}$  and  $\mathcal{P}_1, \dots, \mathcal{P}_L$  form a proof (a sequence of formulas obtained using the rules of inference).

$$\vdash \begin{array}{|l} \mathcal{P}_1 \\ \vdots \\ \mathcal{P}_{L-1} \\ \mathcal{P}_L = \mathcal{Q} \end{array}$$

Here it is easier to assume that proofs are written in a Hilbert-style system, with axioms and rules of inference of the form  $\mathcal{P}_a \rightarrow \mathcal{P}_b$ . There are no assumptions or temporary assumptions as in our natural deduction proofs. To obtain  $\mathcal{P} \vdash \mathcal{Q}$ , we actually prove  $\mathcal{P} \rightarrow \mathcal{Q}$ .

Thus, we can easily check that:

- either  $\mathcal{P}_i$  is an axiom;
- or  $\mathcal{P}_i$  follows from  $\mathcal{P}_a$  and  $\mathcal{P}_b$ , where  $a, b < i$  and  $\mathcal{P}_a$  has the form  $\mathcal{P}_b \rightarrow \mathcal{P}_i$ .
- If  $\mathcal{P}_1, \dots, \mathcal{P}_L$  form a valid proof and  $\mathcal{P}_L = \mathcal{Q}$ , return True.

### 3.3. Numerical value vs. numeral

We return to the difference between a number and its representation, with two new notations:

- $\ulcorner \mathcal{P} \urcorner$ : the Gödel number of  $\mathcal{P}$ ,
- the **numeral**  $\bar{n}$ : the representation of a natural number  $n$  in the language  $\mathcal{L}_0$ .

|                                             |                       |                                                                        |                       |                                                                                                |
|---------------------------------------------|-----------------------|------------------------------------------------------------------------|-----------------------|------------------------------------------------------------------------------------------------|
| $\mathcal{P}$<br>formula of $\mathcal{L}_0$ | $\longleftrightarrow$ | $n = \ulcorner \mathcal{P} \urcorner$<br>Gödel number, in $\mathbb{N}$ | $\longleftrightarrow$ | $\bar{n} = \overline{\ulcorner \mathcal{P} \urcorner}$<br>numeral for $n$ , in $\mathcal{L}_0$ |
|---------------------------------------------|-----------------------|------------------------------------------------------------------------|-----------------------|------------------------------------------------------------------------------------------------|

Example:

$$\mathcal{P} : x = 0 \quad \longleftrightarrow \quad n = \ulcorner \mathcal{P} \urcorner = 2^{\dots} \cdot 3^{\dots} \cdot 5^{\dots} \quad \longleftrightarrow \quad \bar{n} = \overline{\ulcorner \mathcal{P} \urcorner} = \underbrace{SS \dots S}_{n \text{ occurrences}} 0$$

We will often write  $\mathcal{Q}(n)$  instead of  $\mathcal{Q}(\bar{n})$ , since there can be no confusion. If we truly want an algorithm in the language  $\mathcal{L}_0$ ,  $n$  must be written as the numeral  $\bar{n} = SS \dots S0$ .

### 3.4. Concatenation

Two formulas (or words)  $\mathcal{P}$  and  $\mathcal{Q}$  placed side by side give the **concatenation**  $\mathcal{P} * \mathcal{Q}$ . For example, “ $S(x)$ ” \* “ $+$ ” \* “ $y$ ” gives “ $S(x) + y$ ”. This operation seems immediate, but it is more delicate to express when our formulas are represented by natural numbers:

- $\ulcorner S(x) \urcorner = 2^a \cdot 3^b \cdot 5^c \cdot 7^d$

- $\ulcorner + \urcorner = 2^e$
- $\ulcorner y \urcorner = 2^f$
- $\ulcorner S(x) + y \urcorner = 2^a \cdot 3^b \cdot 5^c \cdot 7^d \cdot 11^e \cdot 13^f$

Formally, if  $n = \prod_{i=1}^l \pi_i^{\alpha_i}$  and  $m = \prod_{j=1}^{l'} \pi_j^{\beta_j}$ , then:

$$n * m = n \times \prod_{j=1}^{l'} \pi_{j+l}^{\beta_j}$$

### 3.5. Substitution

Consider a formula  $\mathcal{P}$  with free variable  $x$ , usually denoted by  $\mathcal{P}(x)$ . When we write  $\mathcal{P}(n)$ , this is actually the substitution  $\mathcal{P}[n/x]$ . We must now express this in terms of Gödel numbers. We need a function  $\text{substitution}(\ulcorner \mathcal{P} \urcorner, \ulcorner x \urcorner, n) : \mathbb{N}^3 \rightarrow \mathbb{N}$  that returns  $\ulcorner \mathcal{P}(\bar{n}) \urcorner$ , the Gödel number of the result of the substitution.

#### Example.

- $\mathcal{P}(x) : "y = x + 0"$ , with Gödel number  $\ulcorner \mathcal{P}(x) \urcorner = 2^{\dots} \cdot 3^{\dots} \cdot 5^{\dots} \cdot 7^{\dots} \cdot 11^{\dots}$ .
- $\mathcal{P}(2) : "y = 2 + 0"$ , but "2" is not a symbol of the language; we must write  $\bar{2} = SS0$ .
- Thus,  $\mathcal{P}(\bar{2}) : "y = SS0 + 0"$  is indeed a formula of  $\mathcal{L}_0$ .
- $\ulcorner \mathcal{P}(\bar{2}) \urcorner = 2^{\dots} \cdot 3^{\dots} \cdot 5^{\dots} \cdot 7^{\dots} \cdot 11^{\dots} \cdot 13^{\dots} \cdot 17^{\dots}$  is the Gödel number of the result of the substitution.

Writing the substitution algorithm using concatenation is a good and not too difficult exercise.

## 4. Informal proof of the Incompleteness Theorem

### 4.1. Russell's paradox

We will present the general idea behind the proof of the First Incompleteness Theorem. The proof relies on diagonalization, which here takes the form of self-reference, that is, a formula that speaks about itself. We begin with a short analogy involving Russell's paradox, popularized as the following puzzle:

*In a village, the barber shaves all the men who do not shave themselves, and only those men.*

*Who shaves the barber?*

If we suppose that the barber does not shave himself, then the barber shaves himself. And if he shaves himself, then he must not shave himself. There is no way out!

Russell used the following version to justify that there can be no set containing all sets. If such a set  $\mathcal{E}$  existed, we could consider:

$$y = \{x \in \mathcal{E} \mid x \notin x\}$$

Question: Is  $y \in y$ ?

Remark:  $y$  is indeed a set in  $\mathcal{E}$  because  $\mathcal{E}$  contains all sets.

- If  $y \in y$ , then by the definition of  $y$ ,  $y \notin y$ .
- If  $y \notin y$ , then by the definition of  $y$ ,  $y \in y$ .

This is a contradiction, so the set of all sets cannot exist.

## 4.2. Diagonalization and the Gödel formula

### Starting formula.

Here is the formula we will study:

$\mathcal{P}(x)$ : “The formula obtained from the formula with Gödel number  $x$ , by substituting  $x$  for its free variable, is unprovable.”

Of course, in the following chapters we will write this formula in purely mathematical terms.

For now, observe that:

- A natural number  $n$  can be viewed as the Gödel number of a formula  $\mathcal{Q}$  that we can recover (see the previous sections).
- If this formula  $\mathcal{Q}$  has a free variable, for example  $\mathcal{Q}(y)$ , then we can perform the substitution  $\mathcal{Q}[n/y]$ , denoted by  $\mathcal{Q}(n)$ .
- Finally, for a given  $n$ , we can ask whether the closed formula  $\mathcal{Q}(n)$  is provable. This is exactly what the formula  $\mathcal{P}(n)$  does: “The formula  $\mathcal{Q}(n)$  is unprovable.”

We must now justify that  $\mathcal{P}(x)$  is indeed a formula of our language  $\mathcal{L}_0$ . We saw that “`is_formula()`” and “`is_proof()`” are simple algorithms, which means that they could be expressed in the language  $\mathcal{L}_0$  of Peano arithmetic. But what about the assertion “this formula is provable”?

We can define:

$$\text{is\_provable}(n) \quad \text{iff} \quad \exists N \text{ is\_proof}(N, n).$$

The main problem, from an algorithmic viewpoint, is that we have no a priori bound on  $N$ , so a corresponding algorithm would use a “while” loop to test  $N = 1, N = 2, \dots$ . If the formula is provable, the algorithm eventually stops, but it does not if the formula is unprovable. It is therefore a bad algorithm from our point of view.

On the other hand, “ $\exists N \text{ is\_proof}(N, n)$ ” is indeed a formula of our first-order language. Thus, “this formula is unprovable” is indeed a formula of  $\mathcal{L}_0$ . It follows that  $\mathcal{P}(x)$ , with  $x$  as a free variable, is a formula of  $\mathcal{L}_0$ .

### Diagonalization.

Since  $\mathcal{P}(x)$  is a formula of  $\mathcal{L}_0$ , it has a Gödel number. Let:

$$n = \ulcorner \mathcal{P}(x) \urcorner$$

be the Gödel number of the formula  $\mathcal{P}$ .

The Gödel formula is then defined by:

$$\mathcal{G} = \mathcal{P}(n)$$

It is therefore the formula  $\mathcal{P}$  evaluated at its own Gödel number; in other words:

$$\mathcal{G} = \mathcal{P}(\ulcorner \mathcal{P} \urcorner)$$

### The Incompleteness Theorem.

With this still informal formulation, here are the Incompleteness Theorem and its proof.

**Theorem 2.**

*There exist true formulas that are unprovable.*

We will show that  $\mathcal{G}$  is such a formula. First,  $\mathcal{G}$  says, “the formula obtained from the formula with Gödel number  $n$ , by substituting  $n$  for its free variable, is unprovable”. But “the formula obtained from the formula with Gödel number  $n$ , by substituting  $n$  for its free variable” denotes exactly the formula  $\mathcal{P}$  evaluated at  $n$ , that is,  $\mathcal{G} = \mathcal{P}(n)$ , because  $n$  is the Gödel number of  $\mathcal{P}$ . Thus, the sentence  $\mathcal{G}$  says exactly, “ $\mathcal{G}$  is unprovable”.

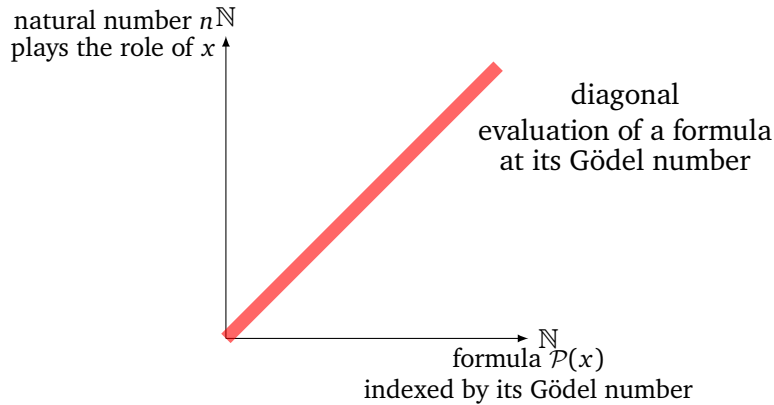
*Proof.*

- Suppose for a contradiction that  $\mathcal{G}$  is false. Then  $\neg\mathcal{G}$  is true, which means “ $\mathcal{G}$  is provable”. But by the Soundness Theorem, if  $\mathcal{G}$  is provable, then  $\mathcal{G}$  is true. This is a contradiction.
- Thus,  $\mathcal{G}$  is true, but this means that “ $\mathcal{G}$  is unprovable”.
- Conclusion:  $\mathcal{G}$  is true but unprovable.

□

### 4.3. Cantor’s diagonal

Why is the procedure explained above called diagonalization? The diagram below makes this easy to understand. We have formulas  $\mathcal{P}$  (or  $\mathcal{P}(x)$ ) that, as we have seen, are indexed by natural numbers in  $\mathbb{N}$  (their Gödel numbers). We also have natural numbers  $n \in \mathbb{N}$  that play the role of  $x$  in the evaluation. The Gödel sentence  $\mathcal{G}$  is obtained by applying the formula  $\mathcal{P}$  precisely at  $n = \ulcorner \mathcal{P} \urcorner$ .



Where does this idea of diagonalization come from? It is already present in Russell’s paradox, where the question “Is  $x \in y$ ?” becomes, on the diagonal, “Is  $y \in y$ ?”. Cantor famously used this method to prove that the set of real numbers is uncountable. Let us study his proof.

**Theorem 3.**

*$\mathbb{R}$  is uncountable.*

*Proof.* Suppose for a contradiction that the set of real numbers  $[0, 1[$  is countable, that is, in bijection with  $\mathbb{N}$ . In more familiar terms, this means that all real numbers in  $[0, 1[$  can be numbered  $x_0, x_1, x_2, \dots$ ; that is,  $[0, 1[ = \{x_k\}_{k \in \mathbb{N}}$ .

Write each real number in  $[0, 1[$  in decimal notation, for example:

$$x_0 = 0.189\,335\dots$$

$$x_1 = 0.772\,004\dots$$

$$x_2 = 0.191\,542\dots$$

$$x_3 = 0.053\,477\dots$$

|          |                |   |   |   |   |   |          |   |   |   |   |   |   |
|----------|----------------|---|---|---|---|---|----------|---|---|---|---|---|---|
| $\vdots$ |                |   |   |   |   |   | $\vdots$ |   |   |   |   |   |   |
| $\vdots$ |                |   |   |   |   |   | $\vdots$ |   |   |   |   |   |   |
| $x_3$    | 0              | 5 | 3 | 4 | 7 | 7 |          | 0 | 5 | 3 | 1 | 7 | 7 |
| $x_2$    | 1              | 9 | 1 | 5 | 4 | 2 |          | 1 | 9 | 2 | 5 | 4 | 2 |
| $x_1$    | 7              | 7 | 2 | 0 | 0 | 4 |          | 7 | 1 | 2 | 0 | 0 | 4 |
| $x_0$    | 1              | 8 | 9 | 3 | 3 | 5 |          | 2 | 8 | 9 | 3 | 3 | 5 |
|          | Decimal digits |   |   |   |   |   |          |   |   |   |   |   |   |

- if this digit is 1, change it to 2;
- if this digit is not 1, change it to 1.

But by construction,  $x'$  cannot be any of the  $x_k$ , because  $x'$  and  $x_k$  have different digits in position  $k$ . This is a contradiction, since  $\{x_k\}_{k \in \mathbb{N}}$  was supposed to enumerate all real numbers in  $[0, 1[$ . Therefore,  $[0, 1[$  is uncountable, and consequently so is  $\mathbb{R}$ .  $\square$

# Primitive Recursive Functions

*We define and study the functions that express the “easy” computations of arithmetic.*

In this chapter, our viewpoint is *semantic*: we study objects from “reality”—natural numbers, prime numbers, addition, and so on.

In the next chapter, we will connect this with the language  $\mathcal{L}_0$  of Peano arithmetic, which provides the *syntactic* viewpoint in which everything is expressed as sequences of symbols. The bridge between the two will be the representability theorem: primitive recursive functions and predicates can be represented in the language of Peano arithmetic.

## 1. Primitive recursive functions

Primitive recursive functions are the functions that are “easy to compute” in the sense of the previous chapter: their values can be computed by an algorithm using only “for” loops.

They are functions  $f : \mathbb{N}^p \rightarrow \mathbb{N}$ . They are defined from three basic functions and two elementary operations. Nevertheless, this class contains all the usual functions and operations on natural numbers: addition, multiplication, exponentiation, and so on.

### 1.1. Definition

We write  $\mathcal{F}_p = \{f : \mathbb{N}^p \rightarrow \mathbb{N}\}$  and  $\mathcal{F} = \bigcup_{p \geq 1} \mathcal{F}_p$ . Recall that  $\mathbb{N}^p = \mathbb{N} \times \cdots \times \mathbb{N}$ . We will often denote the  $p$ -tuple  $(x_1, \dots, x_p) \in \mathbb{N}^p$  by  $\mathbf{x}$ . Thus, the elements of  $\mathcal{F}$  are functions of  $p$  variables (where  $p$  may vary from one function to another).

The definition of recursive functions is based on three basic functions and two operations.

#### Basic functions

- $Z$ : the zero function

$$\begin{aligned} Z : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto 0 \end{aligned}$$

- $S$ : the successor function

$$\begin{aligned} S : \mathbb{N} &\longrightarrow \mathbb{N} \\ x &\longmapsto x + 1 \end{aligned}$$

- $P_i^p$ : the projections (for  $1 \leq i \leq p$ ).

$$\begin{aligned} P_i^p : \mathbb{N}^p &\longrightarrow \mathbb{N} \\ (x_1, \dots, x_p) &\longmapsto x_i \end{aligned}$$

### Composition operation

- Given:  $f_1, \dots, f_q : \mathbb{N}^p \rightarrow \mathbb{N}$  and  $g : \mathbb{N}^q \rightarrow \mathbb{N}$ .
- Composition: the function  $g(f_1, \dots, f_q) : \mathbb{N}^p \rightarrow \mathbb{N}$  defined by

$$\mathbf{x} \mapsto g(f_1(\mathbf{x}), \dots, f_q(\mathbf{x}))$$

### Recursion operation

- Given:
  - $g : \mathbb{N}^p \rightarrow \mathbb{N}$  (base case)
  - $h : \mathbb{N}^{p+2} \rightarrow \mathbb{N}$  (recursion step)
- New function  $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$  defined by:

$$f(\mathbf{x}, 0) = g(\mathbf{x}) \quad \text{and} \quad f(\mathbf{x}, y + 1) = h(\mathbf{x}, y, f(\mathbf{x}, y))$$

This should be viewed as recursion, just as for any sequence defined by an initial value  $u_0$  and a relation  $u_{n+1} = h(n, u_n)$ . The recursion is on the last variable (here  $y$ ).

#### Definition.

The set  $PR$  of **primitive recursive functions** is the smallest subset of  $\mathcal{F}$  that contains  $Z, S, P_i^p$  (for every  $p \geq 1, 1 \leq i \leq p$ ) and is closed under composition and recursion.

What does “being closed” mean? For example, for composition, if  $f_1, \dots, f_q \in PR$  and  $g \in PR$ , then the composite  $g(f_1, \dots, f_q)$  is still in  $PR$ . The same applies to recursion.

Note. The term “primitive recursive” is poorly chosen, but it is the standard terminology. A more evocative name would be “easily computable” function, but that term is used in other situations.

## 1.2. First examples

### (a) Constant functions $\mathbb{N} \rightarrow \mathbb{N}$

Constant functions  $f : \mathbb{N} \rightarrow \mathbb{N}$  are primitive recursive. We know this for the zero function  $Z$ . For the constant 1, we have  $S(Z(x)) = S(0) = 1$ , so  $f = S \circ Z$  is primitive recursive as the composite of two basic functions. More generally,  $x \mapsto k$  belongs to  $PR$  (by composing  $S$  with itself  $k$  times and applying the result to  $Z$ ).

### (b) Constant functions $\mathbb{N}^2 \rightarrow \mathbb{N}$

Let  $k \in \mathbb{N}$  be fixed. Define:

- base case:  $g : \mathbb{N} \rightarrow \mathbb{N}$ , the constant function equal to  $k$ , which belongs to  $PR$ ;
- recursion step:  $h : \mathbb{N}^3 \rightarrow \mathbb{N}$  defined by  $h(x, y, z) = z$ . This is the projection  $P_3^3$ , so it is primitive recursive.

The recursion operation constructs a function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  in  $PR$  such that  $f(x, 0) = k$  and  $f(x, y + 1) = f(x, y)$ . By induction on  $y$ ,  $f(x, y) = k$  for every  $y$ . Thus  $f : \mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto k$  is primitive recursive.

By induction on  $p$ , one could prove that every constant function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is primitive recursive.

### (c) Elementary arithmetic operations

**Proposition 1.**

The following functions are primitive recursive:

- *addition*:  $\mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto x + y$ ;
- *multiplication*:  $\mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto x \times y$ ;
- *exponentiation*:  $\mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto x^y$ .

*Proof.*

- Addition satisfies the recursion relation  $x + (y + 1) = (x + y) + 1$ . For the base case, let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be defined by  $g(x) = x$  (in fact,  $g(x) = P_1^1(x) = x$ , so it is primitive recursive). Next, let  $h : \mathbb{N}^3 \rightarrow \mathbb{N}$  be defined by  $h(x, y, z) = S(z)$  (in fact,  $h = S \circ P_3^3$ , so  $h$  is primitive recursive). The recursion operation yields a primitive recursive function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ .

Let us verify that  $f(x, y) = x + y$ . Fix  $x$  and proceed by induction on  $y$ .

- For  $y = 0$ ,  $f(x, 0) = g(x) = x = x + 0$ .
- Suppose  $f(x, y) = x + y$ . Then:

$$f(x, y + 1) = h(x, y, f(x, y)) = S(f(x, y)) = f(x, y) + 1 = (x + y) + 1 = x + (y + 1)$$

We have proved by induction on  $y$  that  $f(x, y) = x + y$  for all  $x, y$ . We knew that  $f$  was primitive recursive, so addition is as well.

- For multiplication, use the relation  $x \times (y + 1) = x \times y + x$ . This is left as an exercise.
- Exponentiation is also left as an exercise.

□

**(d) Truncated subtraction****Proposition 2.**

- The predecessor function  $\text{prec} : \mathbb{N} \rightarrow \mathbb{N}$ ,

$$\text{prec}(x) = \begin{cases} x - 1 & \text{if } x \geq 1 \\ 0 & \text{if } x = 0 \end{cases}$$

is primitive recursive.

- The **truncated subtraction**  $\mathbb{N}^2 \rightarrow \mathbb{N}, (x, y) \mapsto x \dot{-} y$ , is also primitive recursive.

$$x \dot{-} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

*Proof.*

- The function  $\text{prec} : \mathbb{N} \rightarrow \mathbb{N}$  satisfies the following recursion relation:  $\text{prec}(0) = 0$ , and for  $y \geq 0$ ,  $\text{prec}(y + 1) = y$ . This intuitively makes it primitive recursive. Technically, however, it is a little more complicated because the recursion operation produces a function  $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$ . Proceed as follows:

- $g(x) = Z(x) = 0$  belongs to PR.
- $h(x, y, z) = P_2^3(x, y, z) = y$  also belongs to PR.

Thus, the function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  defined by  $f(x, 0) = 0$  and satisfying  $f(x, y + 1) = h(x, y, f(x, y)) = y$  belongs to  $PR$ . Hence  $f(x, y)$  is primitive recursive.

Note:  $f(x, y)$  is a function of two variables in which  $x$  serves no purpose. We eliminate it through the following composition:  $\text{prec} : \mathbb{N} \rightarrow \mathbb{N}$ , defined by  $\text{prec}(y) = f(P_1^1(y), P_1^1(y)) = f(y, y)$ , which is therefore primitive recursive.

- Truncated subtraction satisfies the recursion relation  $x \dot{-} 0 = x$  and  $x \dot{-} (y + 1) = \text{prec}(x \dot{-} y)$ . Let  $g(x) = P_1^1(x) = x$  and  $h(x, y, z) = \text{prec}(P_3^3(x, y, z)) = \text{prec}(z)$ . Since  $g$  and  $h$  are primitive recursive, the recursion operation yields a primitive recursive function  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  satisfying  $f(x, y) = x \dot{-} y$ .

□

### 1.3. Remarks

- **Another definition of  $PR$ .** We can also define the set of primitive recursive functions “from below”.

$$PR_0 = \{Z, S\} \cup \{P_i^p\}_{p \geq 1, 1 \leq i \leq p}$$

$$PR_{n+1} = PR_n \cup \{\text{functions obtained from } PR_n \text{ by composition}\}$$

$$\cup \{\text{functions obtained from } PR_n \text{ by recursion}\}$$

Then  $PR = \bigcup_{n \geq 0} PR_n$ .

- **Variables.** We can permute the variables however we wish. For example, if  $f : \mathbb{N}^2 \rightarrow \mathbb{N}$ ,  $(x, y) \mapsto f(x, y)$  is primitive recursive, then  $(x, y) \mapsto f(y, x)$  is also primitive recursive. Indeed,  $f(y, x) = f(P_2^2(x, y), P_1^2(x, y))$ . Thus, we often use the variables  $x, y, z$  directly and in any order we wish.
- **Functions  $f : \mathbb{N}^0 \rightarrow \mathbb{N}$ .** The elements of  $\mathbb{N}^p$  are  $p$ -tuples  $(x_1, \dots, x_p)$ . For  $p = 0$ ,  $\mathbb{N}^0$  consists of a single 0-tuple, denoted by  $()$ . Thus  $\mathbb{N}^0 = \{()\}$ , and a function  $f : \mathbb{N}^0 \rightarrow \mathbb{N}$  has only one value (because its domain has only one element), so it is a constant function.

Until now, we excluded the case  $p = 0$ . It can be useful to allow it and to adopt the convention that functions  $f : \mathbb{N}^0 \rightarrow \mathbb{N}$  are primitive recursive (they are regarded as constants).

This convention allows us to use the recursion operation with  $p = 0$ . It simplifies the proof that  $\text{prec}$  is primitive recursive (indeed, the variable  $x$  was unnecessary in the proof). The proof that  $n!$  is primitive recursive by this method is left as an exercise.

### 1.4. Algorithms

Let us connect this with the previous chapter. If  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is primitive recursive, then for every  $\mathbf{x} \in \mathbb{N}^p$ , we can compute  $f(\mathbf{x})$  with an algorithm using only “for” loops (with a well-controlled number of iterations).

To prove that the functions in  $PR$  satisfy a certain property, it suffices to verify it for the three basic functions and show that the property is preserved by composition and recursion.

Clearly,  $Z(n) = 0$ ,  $S(x) = x + 1$ , and  $P_i^p(\mathbf{x}) = x_i$  are easy to compute. If  $f$  is the composite  $g(f_1, \dots, f_q)$  and there are easy algorithms for  $f_1, \dots, f_q$  and  $g$ , then these algorithms compute  $y_1 = f_1(\mathbf{x}), \dots, y_q = f_q(\mathbf{x})$ , followed by  $g(y_1, \dots, y_q)$ , thereby computing  $f(\mathbf{x})$  through a finite sequence of easy algorithms.

If  $f$  is obtained by recursion,  $f(\mathbf{x}, 0) = g(\mathbf{x})$  is assumed to be computable by an easy algorithm. This lets us compute  $f(\mathbf{x}, 1) = h(\mathbf{x}, 0, f(\mathbf{x}, 0))$ , where  $h$  is also computed by an easy algorithm. Step by step, we compute  $f(\mathbf{x}, 2), f(\mathbf{x}, 3), \dots$  up to the desired value.

## 2. Sets and predicates

### 2.1. Sets

#### Definition.

Let  $A \subset \mathbb{N}^p$ . The **characteristic function**  $\chi_A : \mathbb{N}^p \rightarrow \mathbb{N}$  is defined by:

$$\chi_A(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in A \\ 0 & \text{otherwise} \end{cases}$$

It is also sometimes denoted by  $\mathbb{1}_A$ .

#### Definition.

A set  $A \subset \mathbb{N}^p$  is a **primitive recursive set** if its characteristic function  $\chi_A$  is primitive recursive.

For example, the set  $A \subset \mathbb{N}$  of even natural numbers is primitive recursive. Indeed, its characteristic function can be defined by the following recursion relation:

$$\chi_A(0) = 1 \quad \text{and} \quad \chi_A(n+1) = 1 \dot{-} \chi_A(n)$$

This makes  $\chi_A$  primitive recursive.

#### Proposition 3.

If  $A$  and  $B$  are primitive recursive subsets of  $\mathbb{N}^p$ , then  $A \cup B$ ,  $A \cap B$ , and  $\mathbb{N}^p \setminus A$  are also primitive recursive sets.

For the proof, we need the **sign function**  $\text{sgn} : \mathbb{N} \rightarrow \mathbb{N}$  defined by:

$$\text{sgn}(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n > 0 \end{cases}$$

It is primitive recursive because it can also be written as  $\text{sgn}(n) = 1 \dot{-} (1 \dot{-} n)$ .

*Proof.* The characteristic functions of these sets can be expressed using the primitive recursive functions  $\chi_A$  and  $\chi_B$ :

- $\chi_{A \cap B} = \chi_A \times \chi_B$
- $\chi_{A \cup B} = \text{sgn}(\chi_A + \chi_B)$
- $\chi_{\mathbb{N}^p \setminus A} = 1 \dot{-} \chi_A$

□

#### Proposition 4.

If  $f$  is primitive recursive and  $A = \{\mathbf{x} \in \mathbb{N}^p \mid f(\mathbf{x}) = 0\}$ , then  $A$  is a primitive recursive set.

*Proof.* We have  $\chi_A(\mathbf{x}) = 1 \dot{-} f(\mathbf{x})$ .

□

#### Proposition 5 (Definition by cases).

Let  $g, h : \mathbb{N}^p \rightarrow \mathbb{N}$  be primitive recursive functions and let  $A \subset \mathbb{N}^p$  be a primitive recursive set. The function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  defined by:

$$f(\mathbf{x}) = \begin{cases} g(\mathbf{x}) & \text{if } \mathbf{x} \in A \\ h(\mathbf{x}) & \text{otherwise} \end{cases}$$

is primitive recursive.

*Proof.* We can write  $f(\mathbf{x}) = g(\mathbf{x}) \times \chi_A(\mathbf{x}) + h(\mathbf{x}) \times \chi_{\mathbb{N}^p \setminus A}(\mathbf{x})$ . It is expressed using sums and products of primitive recursive functions and is therefore primitive recursive as well.

□

## 2.2. Predicates

### Definition.

A  $p$ -place predicate  $P : \mathbb{N}^p \rightarrow \{T, F\}$  is a **primitive recursive predicate** if the set

$$A = \{\mathbf{x} \in \mathbb{N}^p \mid P(\mathbf{x}) \text{ is true}\}$$

is a primitive recursive set.

In other words, this means that the characteristic function:

$$\chi_P(\mathbf{x}) = \begin{cases} 1 & \text{if } P(\mathbf{x}) \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

is primitive recursive.

### Proposition 6.

If  $P$  and  $Q$  are primitive recursive  $p$ -place predicates, then  $\neg P$ ,  $P \wedge Q$ ,  $P \vee Q$ ,  $P \rightarrow Q$ , and  $P \leftrightarrow Q$  are also primitive recursive.

*Proof.* Let  $A$  and  $B$  be the primitive recursive sets associated with  $P$  and  $Q$ , respectively.

- The set  $\mathbb{N}^p \setminus A$  is primitive recursive, so  $\neg P$  is a primitive recursive predicate. (Another proof:  $\chi_{\neg P}(\mathbf{x}) = 1 \dot{-} \chi_P(\mathbf{x})$ .)
- The set associated with  $P \wedge Q$  is  $A \cap B$ , which is primitive recursive (alternatively,  $\chi_{P \wedge Q}(\mathbf{x}) = \chi_P(\mathbf{x}) \times \chi_Q(\mathbf{x})$ ).
- The set associated with  $P \vee Q$  is  $A \cup B$  (alternatively,  $\chi_{P \vee Q}(\mathbf{x}) = \text{sgn}(\chi_P(\mathbf{x}) + \chi_Q(\mathbf{x}))$ ).
- The connectives  $\rightarrow$  and  $\leftrightarrow$  can be expressed using the preceding connectives.

□

### Proposition 7.

The predicates “ $x > y$ ” and “ $x \geq y$ ” are primitive recursive.

*Proof.* Their characteristic functions are:

- $\chi_{>}(x, y) = \text{sgn}(x \dot{-} y)$
- $\chi_{\geq}(x, y) = \text{sgn}((x + 1) \dot{-} y)$

□

## 2.3. Bounded schemas

Searching for elements defined by a minimum, or by the quantifiers  $\forall$  or  $\exists$ , preserves primitive recursiveness provided the search is bounded. This will be fundamental for the applications to arithmetic discussed immediately below.

### Bounded minimization schema

Let  $P(\mathbf{x}, t)$  be a predicate. Fix  $\mathbf{x}$  and  $y$ . We can consider:

$$\min\{t \leq y \mid P(\mathbf{x}, t) \text{ is true}\}.$$

If the set  $\{t \leq y \mid P(\mathbf{x}, t) \text{ is true}\}$  is empty, by convention we set  $\min = y + 1$ .

If  $A$  is a set, we similarly define:

$$\min\{t \leq y \mid (\mathbf{x}, t) \in A\}$$

**Proposition 8.**

If  $P$  is a primitive recursive predicate, then the function  $f(\mathbf{x}, y)$  defined by:

$$f(\mathbf{x}, y) = \min\{t \leq y \mid P(\mathbf{x}, t) \text{ is true}\}$$

is primitive recursive.

The same holds for a primitive recursive set  $A$ .

**Example.**

Consider the integer square-root function  $\mathbb{N} \rightarrow \mathbb{N}$ , which maps  $y$  to  $\lfloor \sqrt{y} \rfloor$  (for example,  $\lfloor \sqrt{10} \rfloor = 3$ ). In fact,  $\lfloor \sqrt{y} \rfloor$  is the least natural number whose successor has a square greater than  $y$ ; that is:

$$\lfloor \sqrt{y} \rfloor = \min\{t \leq y \mid (t+1)^2 > y\}$$

It is therefore primitive recursive.

*Proof.* Intuitively, for fixed  $\mathbf{x}$  and  $y$ , only finitely many values  $t \leq y$  must be tested, so this is easy to compute algorithmically. Fix  $\mathbf{x}$  and find the recursion formula.

- Base case: ( $y = 0$ ). If  $P(\mathbf{x}, 0)$  is true, set  $g(\mathbf{x}) = 0$ ; otherwise, set  $g(\mathbf{x}) = 1$  (by the convention  $\min = y + 1$ ).
- Recursion formula. We seek  $f(\mathbf{x}, y+1) = \min\{t \leq y+1 \mid P(\mathbf{x}, t)\}$ . A definition by cases gives the formula (see Proposition 5):

$$f(\mathbf{x}, y+1) = \begin{cases} f(\mathbf{x}, y) & \text{if } f(\mathbf{x}, y) \leq y \\ y+1 & \text{if } f(\mathbf{x}, y) = y+1 \text{ and } P(\mathbf{x}, y+1) \text{ is true} \\ y+2 & \text{if } f(\mathbf{x}, y) = y+1 \text{ and } P(\mathbf{x}, y+1) \text{ is false} \end{cases}$$

□

**Bounded quantification****Proposition 9.**

Let  $A \subset \mathbb{N}^{p+1}$  be a primitive recursive set. Then the sets:

- $B = \{(\mathbf{x}, y) \mid \forall t \leq y, (\mathbf{x}, t) \in A\}$
- $C = \{(\mathbf{x}, y) \mid \exists t \leq y, (\mathbf{x}, t) \in A\}$

are also primitive recursive.

*Proof.* Let  $A_t = \{(\mathbf{x}, t) \in A\}$ . Then  $\chi_B = \chi_{A_0} \times \chi_{A_1} \times \cdots \times \chi_{A_y} = \prod_{t=0}^y \chi_{A_t}$  and  $\chi_C = \text{sgn}(\sum_{t=0}^y \chi_{A_t})$ . The bounded sum and product are easily defined by recursion. □

This also works in predicate form. For example, “being a perfect square” is a primitive recursive predicate. Recall that  $y$  is a perfect square if there exists  $t$  such that  $y = t^2$  ( $0, 1, 4, 9, 16, 25, \dots$  are perfect squares). This predicate is defined by bounded quantification:

$$\exists t \leq y, y = t^2.$$

**2.4. Arithmetic**

**Proposition 10.**

The following functions are primitive recursive:

- $q(x, y)$ , the quotient in the Euclidean division of  $x$  by  $y$  (with the convention  $q(x, 0) = x + 1$ );
- $\pi(n)$ , which computes the  $n$ th prime number;
- $\text{exponent}(n, i)$ , which computes the exponent of  $\pi(i)$  in the prime factorization of  $n$ .

**Proposition 11.**

The set  $\mathcal{P}$  of prime numbers is primitive recursive.

We prove these two propositions together.

*Proof.*

- $q(x, y) = \min\{q \leq x \mid (q + 1) \times y > x\}$  is therefore primitive recursive.
- The remainder  $r(x, y) = x \dot{-} (q(x, y) \times y)$  is primitive recursive as well.
- The predicate  $\text{is\_divisible}(x, y)$  (which is true if  $y$  divides  $x$ ) is obtained from  $r(x, y) = 0$ . It is therefore primitive recursive.
- Being prime means being strictly greater than 1 and having no divisors other than 1 and itself. Thus  $\mathcal{P}$  is a primitive recursive set:

$$\mathcal{P} = \{x \mid x > 1\} \cap \{x \mid \forall y \leq x \quad (y = 1 \vee y = x \vee \neg \text{is\_divisible}(x, y))\}$$

Equivalently, the predicate  $\text{is\_prime}(n)$  is primitive recursive.

- The function  $\pi(n)$  that returns the  $n$ th prime number (denoted by  $\pi_n$ , with  $\pi_1 = 2$ ) satisfies the following recursion relation, which makes it primitive recursive:

$$\pi(n + 1) = \min \{t \leq \pi(n)! + 1 \mid t > \pi(n) \wedge \text{is\_prime}(t)\}$$

Recall that there is always a prime number between  $\pi_n$  and  $\pi_n! + 1$  (or even between  $m$  and  $2m$ ).

- The exponent of  $\pi_i$  in the prime factorization of  $n$  is defined by:

$$\text{exponent}(n, i) = \min \{k \leq n \mid \neg \text{is\_divisible}(n, \pi_i^{k+1})\}$$

It is therefore primitive recursive.

□

Let us finish with a famous counterexample.

**Example.**

The Ackermann function  $A : \mathbb{N}^2 \rightarrow \mathbb{N}$  is not primitive recursive. It is defined as follows:

- $A(0, n) = n + 1$
- $A(m + 1, 0) = A(m, 1)$
- $A(m + 1, n + 1) = A(m, A(m + 1, n))$

For fixed  $m$ , the function  $A_m : \mathbb{N} \rightarrow \mathbb{N}, n \mapsto A(m, n)$  is primitive recursive. But the function  $A : \mathbb{N}^2 \rightarrow \mathbb{N}$  is not, because its nested recursion makes it grow faster than any primitive recursive function.

### 3. Application to Gödel coding

We will justify, in the world of natural numbers, that all the operations we wish to perform on Gödel numbers are carried out by primitive recursive functions; in other words, as presented earlier, the computations are performed by algorithms with “for” loops.

#### 3.1. Length, exponent, concatenation, substitution

Consider a Gödel number  $n = \pi_1^{c_1} \times \cdots \times \pi_l^{c_l}$  (with  $c_i \geq 1$ ).

- First, we want to recover  $l$  from  $n$  by a primitive recursive procedure:

$$\text{length}(n) = \min\{i \leq n \mid \pi_{i+1} \text{ does not divide } n\}$$

- We have already studied the function  $\text{exponent}(n, i)$ , which returns the exponent  $c_i$ . Thus, we can decode a Gödel number using a primitive recursive function.
- We saw the concatenation formula in the previous chapter. If  $n = \prod_{i=1}^l \pi_i^{\alpha_i}$  and  $m = \prod_{j=1}^{l'} \pi_j^{\beta_j}$ , then:

$$n * m = n \times \prod_{j=1}^{l'} \pi_{j+l}^{\beta_j}$$

These are finite products, so this is a primitive recursive operation.

- Consider the function  $\text{substitution}(p, v, n)$ : for a formula  $\mathcal{P}$  (given by its Gödel number  $p$ ), with free variable  $x$  (given by its code  $v$ ), it returns the Gödel number of  $\mathcal{P}(n)$  (that is,  $\mathcal{P}(n/x)$ ).

First extract each code  $c_i$  from  $\mathcal{P}$  and compare it with the code of the variable  $x$ . If they are equal, replace that symbol with the term  $n$  using concatenation. This is a sequence of primitive recursive functions, so substitution is primitive recursive.

#### 3.2. Variables, terms, formulas

Deciding whether a natural number  $n$  represents a variable, a term, or a valid formula gives primitive recursive predicates. We refer to the previous chapter without further details.

- $\text{is\_variable}(n)$
- $\text{is\_term}(n)$
- $\text{is\_formula}(n)$

#### 3.3. Peano axioms

We are given a natural number  $n$  (a Gödel number) and must decide whether the formula it encodes is one of the Peano axioms. Consider the first Peano axiom “ $\forall x S(x) \neq 0$ ”, which we would write in  $\mathcal{L}_0$  as “ $\forall x \neg Sx = 0$ ”; denote its Gödel number by  $n_1$ .

To determine whether  $n$  encodes this axiom, it suffices to test whether  $n = n_1$ . We do this for all seven axioms.

The difficulty comes from the induction axioms:

$$\text{Rec}_{\mathcal{P}(x), y} : \mathcal{P}(0) \wedge [\forall y \mathcal{P}(y) \rightarrow \mathcal{P}(S(y))] \rightarrow \forall x \mathcal{P}(x)$$

This is an axiom schema: there is one axiom for each formula  $\mathcal{P}(x)$ . Thus, a priori, there are infinitely many formulas and variables to test. Fortunately, for a given Gödel number  $n$ , if  $x$  is a variable occurring

in the formula encoded by  $n$ , then its code  $v$  is necessarily less than  $n$ . Likewise, if  $\mathcal{P}$  is a subformula of the formula encoded by  $n$ , then its code  $p$  also satisfies  $p \leq n$ .

We can therefore construct a primitive recursive predicate of the form:

$$\begin{aligned} \text{is\_induction\_axiom}(n) = & \exists \mathcal{P} \text{ with code } p \leq n \quad \exists x \text{ with code } v \leq n \quad \exists y \text{ with code } w \leq n \\ & \text{such that is\_formula}(\mathcal{P}) \text{ AND is\_variable}(x) \text{ AND is\_variable}(y) \\ & \text{AND } n \text{ encodes the formula } \mathcal{R}ec_{\mathcal{P}(x),y} \end{aligned}$$

Thus, we can recognize all the Peano axioms by a primitive recursive procedure.

### 3.4. Hilbert-style proofs

In the first two parts of this book, we presented proofs and their rules of inference in natural-deduction notation. As its name suggests, natural-deduction proofs are written in a form close to that of a mathematical proof. However, they have assumptions and temporary assumptions, and this presentation is not suited to algorithmic processing.

We now consider Hilbert-system notation, which is better suited to Gödel coding.

- First, we dispense with assumptions. If we had to prove  $\mathcal{P} \vdash \mathcal{Q}$ , we would prove  $\vdash (\mathcal{P} \rightarrow \mathcal{Q})$ .
- There is only one rule of inference for propositional logic:

$$\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \quad \vdash \quad \mathcal{Q} \quad (\textit{modus ponens})$$

- On the other hand, axioms appear: formulas assumed to be always true (regardless of what we wish to prove).
- For propositional logic, there are three axioms (variants exist; this is the Łukasiewicz variant):
  1.  $\mathcal{P} \rightarrow (\mathcal{Q} \rightarrow \mathcal{P})$  (adding an assumption)
  2.  $(\mathcal{P} \rightarrow (\mathcal{Q} \rightarrow \mathcal{R})) \rightarrow ((\mathcal{P} \rightarrow \mathcal{Q}) \rightarrow (\mathcal{P} \rightarrow \mathcal{R}))$  (distribution of implication)
  3.  $(\neg \mathcal{Q} \rightarrow \neg \mathcal{P}) \rightarrow (\mathcal{P} \rightarrow \mathcal{Q})$  (contrapositive)
- Thus, to check whether a sequence  $\mathcal{P}_1, \dots, \mathcal{P}_l$  of formulas is indeed a proof of  $\mathcal{Q}$ , it suffices that for every  $i$ :
  - either  $\mathcal{P}_i$  is one of the axioms;
  - or  $\mathcal{P}_i$  is obtained by *modus ponens* from preceding lines, that is, there exist  $j, k < i$  such that  $\mathcal{P}_k$  is  $\mathcal{P}_j \rightarrow \mathcal{P}_i$ ;

and finally, we check that  $\mathcal{P}_l = \mathcal{Q}$ .

- For first-order logic, we add the following axioms:
  - $(\forall x \mathcal{P}(x)) \rightarrow \mathcal{P}(t)$
  - $[\forall x (\mathcal{P} \rightarrow \mathcal{Q}(x))] \rightarrow [(\mathcal{P} \rightarrow \forall x \mathcal{Q}(x))]$  if  $\mathcal{P}$  has no free variable  $x$ .
  - $x = x$
  - $x = y \rightarrow (\mathcal{P}(x) \rightarrow \mathcal{P}(y))$
- We also add a second rule of inference, a variant of the  $\forall$ -introduction rule:

$$\text{if } \vdash \mathcal{P} \quad \text{then } \vdash \forall x \mathcal{P}$$

### 3.5. Being a proof

We can now see that checking whether a super-Gödel number  $N$  is a proof of  $n$  is easy using the Hilbert system. Recall that there are no assumptions (to prove  $\Gamma \vdash Q$ , we prove  $\vdash (\Gamma \rightarrow Q)$ ), that the rules of inference are very simple (*modus ponens* and  $\forall$  introduction), and that a proof is merely a linear sequence of formulas.

We thus construct a primitive recursive predicate  $\text{is\_proof}(N, n)$  that checks whether the sequence of formulas  $\mathcal{P}_i$  encoded by the super-Gödel number  $N$  is a valid proof of the formula encoded by the Gödel number  $n$ .

This predicate is constructed as follows (for propositional logic):

- extract the expressions  $\mathcal{P}_i$ ,  $i = 1, \dots, l$ , from  $N$  and the expression  $Q$  from  $n$ ;
- check that these expressions are valid formulas;
- for each  $i = 1, \dots, l$ :
  - check whether  $\mathcal{P}_i$  is an axiom;
  - or whether  $\mathcal{P}_i$  follows by *modus ponens* from preceding lines (that is, whether there exist  $j, k < i$  such that  $\mathcal{P}_k$  is  $\mathcal{P}_j \rightarrow \mathcal{P}_i$ ).
- check whether the last formula  $\mathcal{P}_l$  is indeed  $Q$ .

# Representable Functions

*We will show that Peano arithmetic can express all primitive recursive functions.*

Note: This is a technical chapter. Readers in a hurry may simply focus on the result stated at the beginning. However, it is also a pivotal chapter, since we will connect semantics (the world of natural numbers) with syntax (the language of Peano arithmetic). It also contains one of Gödel's technical achievements: the function  $\beta$ . This function encodes any list of numbers using only two natural numbers.

## 1. A primitive recursive predicate is representable

### 1.1. Representable predicate

For a fixed  $p \geq 1$ , write  $\mathbf{x} = (x_1, \dots, x_p)$ . If  $\mathbf{n} = (n_1, \dots, n_p) \in \mathbb{N}^p$ , write  $\bar{\mathbf{n}} = (\bar{n}_1, \dots, \bar{n}_p)$  for the corresponding sequence of numerals.

#### Definition.

A  $p$ -place predicate  $R(\mathbf{x})$  is **representable** if there exists a formula  $\mathcal{P}(\mathbf{x})$  in the language  $\mathcal{L}_0$  such that, for every  $\mathbf{n} \in \mathbb{N}^p$ :

- if  $R(\mathbf{n})$  is true, then:  $\mathcal{PA} \vdash \mathcal{P}(\bar{\mathbf{n}})$
- if  $R(\mathbf{n})$  is false, then:  $\mathcal{PA} \vdash \neg \mathcal{P}(\bar{\mathbf{n}})$

Remark: the full term is “predicate strongly representable in the theory  $\mathcal{PA}$  of Peano arithmetic”.

Notice carefully the passage from a natural number to its numeral. For example, if  $p = 1$ , the predicate is evaluated as  $R(n)$  (with  $n \in \mathbb{N}$ ), while the formula  $\mathcal{P}$  is evaluated as  $\mathcal{P}(\bar{n})$ , where the numeral  $\bar{n} = SSS \dots S0$  represents  $n$  in  $\mathcal{L}_0$ .

#### Example.

The predicate  $R(x)$ : “ $x$  is an even natural number” is representable by the following formula:

$$\mathcal{P}(x) : \exists y \ x = \bar{2} \times y$$

Checking this property for a specific natural number is a finite process that can be translated step by step into a proof within  $\mathcal{PA}$ .

## 1.2. Theorem

Here is the theorem that will be useful in the following chapters. It will be applied to the predicate “is\_proof( $N, n$ )”, which determines whether the formulas encoded by the super-Gödel number  $N$  form a proof of the formula with Gödel number  $n$ .

### Theorem 1.

*If  $R$  is a primitive recursive predicate, then  $R$  is representable.*

Recall that a predicate is primitive recursive if its characteristic function is primitive recursive. Similarly, a predicate will be representable if its characteristic function is representable. We will therefore define and study the more general notion of a representable function, which is essential for proving the theorem above.

## 2. Representable functions

### 2.1. Definition

#### Definition.

A function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is **representable** if there exists a formula  $\mathcal{P}(\mathbf{x}, y)$  such that, for every  $\mathbf{n} = (n_1, \dots, n_p) \in \mathbb{N}^p$ :

$$\mathcal{P}\mathcal{A} \vdash \forall y \left[ \mathcal{P}(\bar{\mathbf{n}}, y) \leftrightarrow y = \overline{f(\mathbf{n})} \right]$$

Remarks:

- The full term is “function strongly representable in the theory  $\mathcal{P}\mathcal{A}$ ”.
- Once again, this definition connects a natural number  $k \in \mathbb{N}$  with its numeral  $\bar{k}$  in  $\mathcal{L}_0$ .
- The formula  $\mathcal{P}$  should be understood as the graph of  $f$ . For the graph of a function from  $\mathbb{R}$  to  $\mathbb{R}$ , we have  $y = f(x)$  if and only if  $(x, y)$  belongs to the graph of  $f$ . Here, for a function from  $\mathbb{N}^p$  to  $\mathbb{N}$ , we have  $y = f(\mathbf{x})$  if and only if  $\mathcal{P}(\mathbf{x}, y)$ . This is an equivalent way to define the function.

#### Example.

The function  $f : \mathbb{N} \rightarrow \mathbb{N}$  defined by  $f(x) = 2x + 1$  is representable. Indeed, consider the formula:

$$\mathcal{P}(x, y) : y = 2x + 1$$

It is therefore true (and provable in Peano arithmetic) that  $\mathcal{P}(x, y)$  holds if and only if  $y = f(x)$ . In other words, for every  $n \in \mathbb{N}$ ,  $\mathcal{P}(n, y)$  is true if and only if  $y = f(n)$ .

But be careful: we have not quite constructed the expression in the language  $\mathcal{L}_0$ . Indeed, the natural number 2 does not exist there as a symbol; it must be written  $\bar{2} = SS0$ . The correct formula in  $\mathcal{L}_0$  is therefore:

$$\mathcal{P}(x, y) : y = S(SS0 \times x)$$

And for every  $n \in \mathbb{N}$ , the equivalence is:

$$\forall y \left[ \mathcal{P}(\bar{n}, y) \leftrightarrow y = \overline{f(n)} \right]$$

(where  $\bar{n} = SS \dots S0$  is the numeral for the natural number  $n$  and  $\overline{f(n)} = SS \dots S0$  is the numeral for the natural number  $f(n)$ ).

This is indeed a passage from the world of natural numbers to the language  $\mathcal{L}_0$ .

## 2.2. Equivalent definition

To better understand the definition of a representable function, here is an equivalent definition.

### Definition.

A function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is **representable** if there exists a formula  $\mathcal{P}(\mathbf{x}, y)$  such that, for every  $\mathbf{n} \in \mathbb{N}^p$ , setting  $m = f(\mathbf{n})$ , we have:

1.  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}(\bar{\mathbf{n}}, \bar{m})$  **correctness**
2.  $\mathcal{P}\mathcal{A} \vdash \forall y [\mathcal{P}(\bar{\mathbf{n}}, y) \rightarrow y = \bar{m}]$  **uniqueness**

Correctness means that the formula  $\mathcal{P}$  does recover the value  $m$  of  $f(\mathbf{n})$ . Uniqueness is essential, since a function can of course take only one value.

Let us show that this second definition is equivalent to the first. The original definition contains an equivalence. Uniqueness is exactly the implication  $\rightarrow$ . Starting from the original definition and specializing the  $\forall$  to  $y = \bar{m}$  gives exactly  $\mathcal{P}(\bar{\mathbf{n}}, \bar{m})$ . Conversely, starting from the second definition, we have  $\mathcal{P}(\bar{\mathbf{n}}, \bar{m})$ , which implies that  $y = \bar{m} \rightarrow \mathcal{P}(\bar{\mathbf{n}}, y)$ . We then introduce  $\forall y$  to obtain the reverse implication ( $\leftarrow$ ).

In the exercises, we will see counterexamples of formulas that fail correctness or uniqueness.

## 2.3. Representable predicate

Let us return to the notion of a representable predicate.

### Proposition 1.

A predicate  $R(\mathbf{x})$  is representable if and only if its characteristic function  $\chi_R(\mathbf{x})$  is representable.

Recall that, by definition,  $\chi_R(\mathbf{x}) = 1$  if  $R(\mathbf{x})$  is true, and  $\chi_R(\mathbf{x}) = 0$  otherwise.

To simplify the proofs, here and often below, we will write them using as few variables as possible, for example with  $p = 1$ . To obtain the general proof, it would of course be enough to replace  $x$  by  $\mathbf{x}$  and  $n$  by  $\mathbf{n}$ . We will also write  $n$  for both the natural number and its numeral  $\bar{n}$ , since there is no ambiguity. The following proof is fairly technical and may be skipped on a first reading.

*Proof.*

•  $\Rightarrow$

Assumption:  $\chi_R$  is a representable function.

Goal:  $R$  is a representable predicate.

Let  $\mathcal{P}(x, y)$  be the formula provided by the assumption, and define  $\mathcal{P}_1(x)$  as  $\mathcal{P}(x, 1)$ . In the formula obtained from the assumption “ $\chi_R$  is a representable function”, set  $y = 1$  to obtain the equivalence:

$$\mathcal{P}\mathcal{A} \vdash [\mathcal{P}(n, 1) \leftrightarrow 1 = \chi_R(n)]$$

- If  $R(n)$  is true, then  $\chi_R(n) = 1$ . Thus  $1 = \chi_R(n)$  is true. By the equivalence,  $\mathcal{P}_1(n)$  is also true. More precisely,  $\mathcal{P}\mathcal{A} \vdash (\mathcal{P}_1(n) \leftrightarrow (1 = 1))$ . Therefore, by the definition of implication, we obtain:  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_1(n)$ .
- If  $R(n)$  is false, then  $\chi_R(n) = 0$ , so the equivalence shows that  $\neg\mathcal{P}_1(n)$  is true.

Thus, the predicate  $R$  is indeed represented by the formula  $\mathcal{P}_1$ .

•  $\Leftarrow$

Assumption:  $R$  is a representable predicate.

Goal:  $\chi_R$  is a representable function.

The assumption gives us a formula  $\mathcal{P}_0(x)$  associated with the predicate  $R$ . We then construct the formula  $\mathcal{P}(x, y)$  for the function  $\chi_R$  as follows:

$$\mathcal{P}(x, y) : (\mathcal{P}_0(x) \wedge (y = 1)) \vee (\neg \mathcal{P}_0(x) \wedge (y = 0))$$

We must verify that this formula does make  $\chi_R$  a representable function.

**Correctness.** Fix a natural number  $n$  and set  $m = \chi_R(n)$ , which is therefore either 0 or 1.

- If  $m = 1$ , then  $R(n)$  is true, so by assumption  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(n)$ . Therefore  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(n) \wedge (m = 1)$ , that is,  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}(n, m)$ .
- If  $m = 0$ , then  $R(n)$  is false, and similarly we obtain  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}(n, m)$ .

In both cases, the correctness condition is satisfied.

**Uniqueness.** For uniqueness, we must show that, for every  $y$ :  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}(n, y) \rightarrow y = m$ .

- Case  $y \neq 0$  and  $y \neq 1$ . Then  $\mathcal{P}(n, y)$  is false, so the implication holds.
- Case  $y = 1$ .
  - \* If  $R(n)$  is true, then  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(n)$ . Therefore  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(n) \wedge (y = 1)$ . Thus  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(n) \wedge (y = 1) \wedge (y = \chi_R(n))$ , so  $\mathcal{P}\mathcal{A} \vdash (\mathcal{P}(n, 1) \rightarrow y = m)$  (because the statements on both sides of the implication are true).
  - \* If  $R(n)$  is false, then  $\mathcal{P}\mathcal{A} \vdash \neg \mathcal{P}_0(n)$ , so  $\mathcal{P}(x, y)$  is false and the implication is true.
- Case  $y = 0$ . This is similar to the preceding case.

□

## 2.4. Important examples

The goal of the chapter is to prove that all primitive recursive functions are representable. We have already encountered the strategy: it is enough to prove this for the three basic functions and then show that the property is preserved under composition and recursion.

### Proposition 2.

*The basic functions  $Z$ ,  $S$ , and the projections  $P_i^p$  are representable functions.*

*Proof.* There is not much to prove. We simply need to provide a suitable formula  $\mathcal{P}(\mathbf{x}, y)$ .

- Consider the successor function  $S$ , defined by  $S(n) = n + 1$ . Set  $\mathcal{P}(x, y) : y = S(x)$  (which is a formula in  $\mathcal{L}_0$ ).

For a fixed  $n \in \mathbb{N}$ , writing  $m = S(n) = n + 1$ , we have:

$$\mathcal{P}(\bar{n}, y) \leftrightarrow y = S(\bar{n}) \leftrightarrow y = \overline{S(n)} \leftrightarrow y = \bar{m}$$

These equivalences are true in any logical system; the Peano axioms are not needed. Thus  $S$  is a representable function.

- For the zero function  $Z$ , the formula is  $\mathcal{P}(x, y) : y = 0$ .
- For the projection  $P_i^p$ , the formula is  $\mathcal{P}(\mathbf{x}, y) : y = x_i$ .

□

**Proposition 3.**

Let  $f_1, \dots, f_q : \mathbb{N}^p \rightarrow \mathbb{N}$  and  $g : \mathbb{N}^q \rightarrow \mathbb{N}$  be representable functions. Then  $h = g(f_1, \dots, f_q)$  is also representable.

Thus representability is preserved under composition. If we prove the same for recursion, we will have the desired result. However, this latter preservation property will be much more difficult to establish and will occupy us for the rest of the chapter.

For now, we focus on the proof of the proposition above. As already mentioned, we will give the proof in a minimal setting, but it is worth discussing the general formula that makes  $h$  a representable function:

$$\mathcal{P}_h(\mathbf{x}, y) : \exists w_1 \dots \exists w_q \quad \mathcal{P}_0(w_1, \dots, w_q, y) \wedge \mathcal{P}_1(\mathbf{x}, w_1) \wedge \dots \wedge \mathcal{P}_q(\mathbf{x}, w_q)$$

where  $\mathcal{P}_0$  is the formula associated with the representable function  $g$ , and  $\mathcal{P}_i$  is the formula for  $f_i$  ( $i = 1, \dots, q$ ). The role of the variables  $w_i$  is particularly interesting: they store the intermediate result  $w_i = f_i(\mathbf{n})$  so that it can be passed to  $g$  to compute  $h(\mathbf{n}) = g(w_1, \dots, w_q) = g(f_1(\mathbf{n}), \dots, f_q(\mathbf{n}))$ .

*Proof.* We prove only the case  $p = 1$  and  $q = 1$ . Let  $f_1 : \mathbb{N} \rightarrow \mathbb{N}$  be a representable function, represented by the formula  $\mathcal{P}_1(x, w)$ . Let  $g : \mathbb{N} \rightarrow \mathbb{N}$  be a representable function, represented by the formula  $\mathcal{P}_0(w, y)$ . We want to show that the composition  $h = g \circ f_1 : \mathbb{N} \rightarrow \mathbb{N}$  is representable.

Define the formula:

$$\mathcal{P}_h(x, y) : \exists w \quad \mathcal{P}_0(w, y) \wedge \mathcal{P}_1(x, w)$$

which will represent the function  $h$ . For  $n \in \mathbb{N}$ , set  $k = f_1(n)$  and  $m = g(k) = g(f_1(n)) = h(n)$ .

- **Correctness.** By assumption,  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_0(k, m)$  and  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_1(n, k)$ . Therefore  $\mathcal{P}\mathcal{A} \vdash \exists w (\mathcal{P}_0(w, m) \wedge \mathcal{P}_1(n, w))$  (it is enough to take  $w = k$ ). Hence  $\mathcal{P}\mathcal{A} \vdash \mathcal{P}_h(n, m)$ .
- **Uniqueness.** By assumption,  $\mathcal{P}\mathcal{A}$  proves the following two implications:

$$\forall w \quad (\mathcal{P}_1(n, w) \rightarrow w = k)$$

$$\text{and } \forall y \quad (\mathcal{P}_0(k, y) \rightarrow y = m)$$

Now suppose that  $\mathcal{P}_h(n, y)$  holds. Then there exists  $w$  such that  $\mathcal{P}_0(w, y)$  and  $\mathcal{P}_1(n, w)$ . Since  $\mathcal{P}_1(n, w)$  holds, we have  $w = k$ . Therefore  $\mathcal{P}_0(k, y)$  holds, which implies  $y = m$ . Thus the uniqueness condition is satisfied:  $\mathcal{P}_h(n, y) \rightarrow y = m$ .

□

**Proposition 4.**

The remainder function, which maps  $a$  and  $b$  to the remainder  $\text{remainder}(b, a)$  of the Euclidean division of  $b$  by  $a$ , is representable.

*Proof.* The formula is:

$$\mathcal{P}(a, b, r) : [a = 0 \wedge b = r] \vee [\exists q \leq b \quad (b = qa + r) \wedge (r < a)]$$

We prove correctness and uniqueness. Indeed, Peano arithmetic proves the existence and uniqueness of Euclidean division. In fact, this would also hold in Robinson arithmetic  $\mathcal{Q}$ , without using induction.

Remark: “ $\exists q \leq b \quad \mathcal{P}$ ” means “ $\exists q \quad (q \leq b) \wedge \mathcal{P}$ ”.

□

### 3. Gödel's $\beta$ function

The goal of this section is to study Gödel's  $\beta$  function. This ingenious construction stores any list of natural numbers  $(n_1, \dots, n_l)$  using only two natural numbers  $a$  and  $b$ .

This construction acts as memory. Indeed, to compute a sequence recursively, we must compute  $u_1, u_2, \dots, u_{l-1}$  before we can compute  $u_l$ .

#### 3.1. Theorem

##### Theorem 2.

There exists a function  $\beta : \mathbb{N}^3 \rightarrow \mathbb{N}$  that is primitive recursive and representable, and such that, for every natural number  $l$  and every sequence  $(n_1, \dots, n_l)$  of natural numbers, there exist  $a, b \in \mathbb{N}$  such that:

$$\beta(a, b, i) = n_i \quad \text{for } i = 1, \dots, l.$$

The function  $\beta$  is ultimately quite simple, since  $\beta(a, b, i)$  will be the remainder of the Euclidean division of  $b$  by  $a \times i + 1$ . The difficult part is explaining the existence of these  $a$  and  $b$ . For this, we will need a well-known theorem from arithmetic.

#### 3.2. Chinese Remainder Theorem

##### Theorem 3 (Chinese Remainder Theorem).

Let  $a_1, \dots, a_l$  be pairwise coprime positive integers. Let  $n_1, \dots, n_l$  be arbitrary natural numbers. There exists  $x \in \mathbb{N}$  such that, for every  $i = 1, \dots, l$ :

$$x \equiv n_i \pmod{a_i}$$

For two natural numbers, the statement is as follows:

##### Theorem 4.

Let  $a$  and  $b$  be coprime positive integers. For every  $\alpha, \beta \in \mathbb{N}$ , there exists  $x \in \mathbb{N}$  such that:

$$\begin{cases} x \equiv \alpha \pmod{a} \\ x \equiv \beta \pmod{b} \end{cases}$$

From an algebraic viewpoint, we even have a group isomorphism:

$$\mathbb{Z}/ab\mathbb{Z} \simeq \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$$

*Proof.* Define:

$$\phi : \mathbb{Z}/ab\mathbb{Z} \longrightarrow \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$$

by  $\phi(\bar{x}^{ab}) = (\bar{x}^a, \bar{x}^b)$ , where  $\bar{x}^m$  denotes the residue class of the natural number  $x$  modulo  $m$ .

- $\phi$  is well-defined. Indeed, if  $\bar{x}^{ab} = \bar{y}^{ab}$ , then  $x = y + kab$ , so  $\bar{x}^a = \bar{y}^a$  and  $\bar{x}^b = \bar{y}^b$ . Therefore  $\phi(\bar{x}^{ab}) = \phi(\bar{y}^{ab})$ .
- $\phi$  is a group homomorphism:  $\phi(\bar{x}^{ab} + \bar{y}^{ab}) = \phi(\bar{x}^{ab}) + \phi(\bar{y}^{ab})$ .
- $\phi$  is injective. If  $\phi(\bar{x}^{ab}) = (\bar{0}^a, \bar{0}^b)$ , then  $\bar{x}^a = \bar{0}^a$ , so  $x = k_1a$ , and  $\bar{x}^b = \bar{0}^b$ , so  $x = k_2b$ . Thus  $k_1a = k_2b$ . In particular,  $b$  divides  $k_1a$ . By assumption,  $a$  and  $b$  are coprime, so Euclid's lemma implies that  $b$  divides  $k_1$ , and hence  $k_1 = k_3b$ . Thus  $x = k_1a = k_3ba$ . Therefore  $x$  is a multiple of  $ab$ , and the class  $\bar{x}^{ab}$  is  $\bar{0}^{ab}$ . Hence the map  $\phi$  is injective.
- Since the domain and codomain have the same cardinality (namely  $ab$ ), the map is bijective.

- Thus, for a given  $(\alpha, \beta)$ , there exists a preimage  $x$  under  $\phi$ , which says exactly that:

$$\begin{cases} x \equiv \alpha \pmod{a} \\ x \equiv \beta \pmod{b} \end{cases}$$

□

### 3.3. Proof

**Construction of  $a$  and  $b$ .** Let  $n_1, \dots, n_l \in \mathbb{N}$  be given. Choose  $m \in \mathbb{N}$  such that  $m \geq l$  and  $m! \geq n_i$  ( $i = 1, \dots, l$ ). Set  $a = m!$ . The other natural number  $b$  will be constructed using the Chinese Remainder Theorem.

- The natural numbers  $ai + 1$  are pairwise coprime. Indeed, if a prime number  $p$  divides both  $ai + 1$  and  $aj + 1$  (with  $i < j$ ), then, by subtraction,  $p|a(j - i)$ . Thus  $p|a$  or  $p|(j - i)$ . Now  $j - i < l \leq m$  and  $a = m!$ , so in either case  $p \leq m$ . It follows that  $p|m!$  (that is,  $p|a$ ), but since  $p|ai + 1$ , we then have  $p|1$ . This is a contradiction.
- We have  $ai + 1 > n_i$  because  $n_i \leq m! = a$  ( $i = 1, \dots, l$ ).
- By the Chinese Remainder Theorem, there exists  $b \in \mathbb{N}$  such that:

$$b \equiv n_i \pmod{ai + 1} \quad \text{for every } i = 1, \dots, l$$

**Construction of  $\beta$ .** This justifies the construction of  $\beta$ :  $\beta(a, b, i)$  is the remainder of the Euclidean division of  $b$  by  $ai + 1$ . Indeed, this remainder is characterized by  $b \equiv n_i \pmod{ai + 1}$  and  $n_i < ai + 1$ , as above.

**$\beta$  is primitive recursive.** The first part ensures that this coding exists, even though it is somewhat complicated. Decoding, however, remains simple:

$$\beta(a, b, i) = \text{remainder}(b, ai + 1)$$

This remainder is indeed given by a primitive recursive function. In fact:

$$\text{remainder}(b, a) = \min\{r \leq b \mid \exists q \leq b \quad b = qa + r\}$$

The condition “ $\exists q \leq b \quad b = qa + r$ ” is a primitive recursive predicate because the quantification is bounded. The remainder is therefore primitive recursive as well, since it is obtained by bounded minimization. The bound on the minimum is not the natural one, since a Euclidean division generally has  $r < a$ , but the bound  $b$  also covers the exceptional case  $a = 0$ .

**$\beta$  is representable.** The formula to consider is:

$$\mathcal{B}(a, b, i, r) : (\exists q \leq b \quad b = q(ai + 1) + r) \wedge (r < ai + 1)$$

In other words,  $r$  is defined as the remainder of the Euclidean division of  $b$  by  $ai + 1$ . We have seen that the remainder function is representable.

Abuse of notation:  $a, b, i, r$  are not variables of the language  $\mathcal{L}_0$ ; they should be replaced by  $x_1, x_2, \dots$ , but the formulas would become less readable.

**Note.** The advantage of coding with the function  $\beta$  is that it uses only addition and multiplication, which are the only operations allowed by the language  $\mathcal{L}_0$ .

Otherwise, to encode a sequence of natural numbers, we could have used the prime-number coding  $2^{n_1}3^{n_2}5^{n_3}\dots$ . But in  $\mathcal{L}_0$ , we are not yet allowed to compute powers: for example, to compute  $x^n$ , we successively compute  $x, x^2, x^3 \dots$  up to  $x^n$ . We therefore need recursion; in other words, we must be able to store intermediate results. But we do not yet know that recursion preserves representability, since that is exactly what we want to prove.

Once we have established the representability of primitive recursive functions using the function  $\beta$ , we will be able to use (finite) lists of numbers for Gödel coding in the language  $\mathcal{L}_0$ .

## 4. Primitive recursive functions are representable

Let us now tackle the final goal of the chapter.

### Theorem 5.

*Every primitive recursive function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is representable.*

### 4.1. Recursion

To prove that primitive recursive functions are representable, it remains to show that representability is preserved under recursion.

### Proposition 5.

*Let  $g : \mathbb{N}^p \rightarrow \mathbb{N}$  and  $h : \mathbb{N}^{p+2} \rightarrow \mathbb{N}$  be representable functions. The function  $f : \mathbb{N}^{p+1} \rightarrow \mathbb{N}$  defined by the initial function  $g$  and the recursion relation  $h$  is representable.*

This proposition implies the theorem above, since we have already shown that the three basic functions  $Z$ ,  $S$ , and  $P_i^p$  are representable and that composition preserves representability.

### 4.2. The formula

We will prove the case  $p = 1$  to keep the notation as simple as possible.

- $g : \mathbb{N} \rightarrow \mathbb{N}$  is represented by  $\mathcal{P}_g(x, u)$ .
- $h : \mathbb{N}^3 \rightarrow \mathbb{N}$  is represented by  $\mathcal{P}_h(x, y, u, v)$ .
- $f : \mathbb{N}^2 \rightarrow \mathbb{N}$  is defined by  $f(x, 0) = g(x)$  and  $f(x, y + 1) = h(x, y, f(x, y))$ .

The goal is to find a formula that represents  $f$ . The formula  $\mathcal{P}_f(x, y, z)$  says: “there exists a finite sequence (encoded by natural numbers  $a$  and  $b$ ) whose first term is  $z_0 = g(x)$ , each transition satisfies the recursion relation  $z_{i+1} = h(x, i, z_i)$ , and whose last term is  $z_y = z$ ”.

We must translate this sentence into  $\mathcal{L}_0$ . To do so, we use Gödel’s  $\beta$  function. We make a minor modification to obtain convenient indexing: we now assume that  $\beta(a, b, i)$  encodes a sequence  $n_0, n_1, n_2, \dots$  (the indices start at 0). We have seen that the function  $\beta$  is representable; denote the corresponding formula by  $\mathcal{B}(a, b, i, r)$ . Recall that the role of the function  $\beta$  is to store a list of numbers, which here will be the terms  $z_i$  of the sequence.

Here is the formula  $\mathcal{P}_f$  in the language  $\mathcal{L}_0$ :

$$\mathcal{P}_f(x, y, z) : \exists a \exists b \text{ Init}(a, b, x) \wedge \text{Trans}(a, b, x, y) \wedge \mathcal{B}(a, b, y, z)$$

Fix  $x$  (in fact, this variable  $x$  serves no purpose), set  $z_0 = g(x)$ ,  $z_1 = h(x, 0, z_0)$ ,  $z_2 = h(x, 1, z_1), \dots$ , and aim to compute  $z_y$  in order to verify that it is indeed the value  $z = f(x, y)$ .

With  $\mathcal{B}$  denoting the formula associated with Gödel’s  $\beta$  function, the three parts of the formula  $\mathcal{P}_f$  are defined as follows:

- $\text{Init}(a, b, x) : \exists u \ B(a, b, 0, u) \wedge \mathcal{P}_g(x, u)$

Thus  $u = \beta(a, b, 0)$  (the first term in memory) and at the same time  $u = g(x)$ , so  $z_0 = g(x) = \beta(a, b, 0)$ .

- $\text{Trans}(a, b, x, y) : \forall i < y \ \exists u \exists v \ B(a, b, i, u) \wedge B(a, b, i + 1, v) \wedge \mathcal{P}_h(x, i, u, v)$

The conjunctions give  $u = \beta(a, b, i)$ ,  $v = \beta(a, b, i + 1)$ , and  $v = h(x, i, u)$ . Thus, if  $z_i = \beta(a, b, i) = u$ , then  $z_{i+1} = h(x, i, z_i) = h(x, i, u) = v = \beta(a, b, i + 1)$ . By induction, we therefore have  $z_i = \beta(a, b, i)$  for every  $i \leq y$ .

- $B(a, b, y, z)$  means that  $z = \beta(a, b, y)$ , that is,  $z = z_y$ .

### 4.3. Correctness

Consider  $n$  (for the variable  $x$ ),  $l$  (for the variable  $y$ ), and  $m = f(n, l)$  (for the variable  $z$ ). Correctness means that  $\mathcal{PA} \vdash \mathcal{P}_f(n, l, m)$ .

This time, set  $z_0 = g(n)$  and  $z_{i+1} = h(n, i, z_i)$  for  $i = 0, \dots, l - 1$ . Final goal: show that  $m = z_l$ .

We know that there exist  $a, b$  encoding the sequence and satisfying  $\beta(a, b, i) = z_i$  for  $i = 0, \dots, l$ .

- We have  $\beta(a, b, 0) = z_0$ , and therefore  $\mathcal{PA} \vdash B(a, b, 0, z_0) \wedge \mathcal{P}_g(n, z_0)$ . Thus, setting  $u = z_0$ , we obtain  $\mathcal{PA} \vdash \text{Init}(a, b, n)$ .
- Next, for  $z_{i+1}$ , set  $u = z_i$  and  $v = z_{i+1}$ . We have:

$$B(a, b, i, z_i) \wedge B(a, b, i + 1, z_{i+1}) \wedge \mathcal{P}_h(n, i, z_i, z_{i+1})$$

This holds for  $i = 0, \dots, l - 1$ , so  $\mathcal{PA} \vdash \text{Trans}(a, b, n, l)$ .

- Finally,  $z_l = \beta(a, b, l) = f(n, l) = m$ , and hence  $B(a, b, l, m)$  holds.

### 4.4. Uniqueness

We must prove that:

$$\mathcal{PA} \vdash \forall z \ \mathcal{P}_f(n, l, z) \rightarrow z = m$$

that is,  $z = z_l$ .

By induction on  $l$ :

- Base case 0: The formula  $\mathcal{P}_f(n, 0, z)$  (with  $m = f(n, 0) = g(n) = z_0$ ) reduces to:

$$\exists a \exists b \ \text{Init}(a, b, n) \wedge B(a, b, 0, z)$$

(because the condition “ $\forall i < 0$ ” is empty, so the predicate Trans is true). Therefore  $z_0 = \beta(a, b, 0) = z$ , and hence  $z = m$ , as desired.

- **Inductive step.** Fix  $i$  such that  $0 \leq i < l$  and assume uniqueness holds at rank  $i$ :

$$\forall u \ \mathcal{P}_f(n, i, u) \rightarrow u = z_i$$

Let  $z$  be such that  $\mathcal{P}_f(n, i + 1, z)$  holds. There exist witnesses  $a, b$  satisfying  $\text{Init}(a, b, n)$ ,  $\text{Trans}(a, b, n, i + 1)$ , and  $B(a, b, i + 1, z)$ . Restricting  $\text{Trans}(a, b, n, i + 1)$  to indices  $k < i$  shows that these same witnesses  $a, b$  along with the value  $u = \beta(a, b, i)$  satisfy  $\mathcal{P}_f(n, i, u)$ . The induction hypothesis then yields  $u = z_i$ .

The transition step at rank  $i$  yields  $v = \beta(a, b, i + 1)$  such that  $\mathcal{P}_h(n, i, u, v)$ . By representability of  $h$ , we have  $v = h(n, i, u) = h(n, i, z_i) = z_{i+1}$ . Finally, since  $B(a, b, i + 1, z)$  requires  $z = \beta(a, b, i + 1) = v$ , we conclude that  $z = z_{i+1}$ .

- By the induction principle (in  $\mathcal{PA}$ ), the statement is true for every  $i \leq l$ , and uniqueness is proved.

**Remark.** It is possible to do everything in Robinson arithmetic  $\mathcal{Q}$ , because our induction is in fact limited to a finite number of steps (since  $l$  is fixed).

This completes the proof that, for primitive recursive functions, recursion preserves representability.

#### 4.5. Application to predicates

We knew that the three basic primitive recursive functions were representable and that composition preserved representability. We have just completed the difficult proof that representability is also preserved under recursion. By the definition of a primitive recursive function, we deduce Theorem 5: “All primitive recursive functions are representable.”

In particular, we know that a predicate is representable if and only if its characteristic function is representable. We therefore obtain Theorem 1, announced at the very beginning: “Every primitive recursive predicate is representable.”

# Diagonalization

*We bring together all the tools developed so far to prove the crucial step toward incompleteness: the diagonal lemma.*

## 1. Previously on...

The final goal is now close at hand: proving the Incompleteness Theorem. This is another technical chapter, centered on a crucial lemma that requires all the preceding chapters. Before diving into the details, let us briefly review all the concepts we will need.

### 1.1. Peano arithmetic

- Peano arithmetic is based on the language  $\mathcal{L}_0 = (0, S, +, \times)$ , in which we construct a theory  $\mathcal{PA}$  (the Peano axioms).
- The natural numbers  $\mathbb{N}$  form a model of this theory: the Peano axioms are satisfied in  $\mathbb{N}$ .
- For  $n \in \mathbb{N}$ , its numeral  $\bar{n} = SS \dots S0$  is its representation in  $\mathcal{L}_0$  (there are exactly  $n$  symbols  $S$ ).

### 1.2. Gödel coding

- Every formula of  $\mathcal{L}_0$  is assigned a natural number: its Gödel number. We also write it as  $n = \ulcorner \mathcal{P} \urcorner$ .
- In  $\mathcal{L}_0$ , its numeral is therefore  $\bar{n} = \overline{\ulcorner \mathcal{P} \urcorner}$ , which we will also often denote by  $\ulcorner \mathcal{P} \urcorner$  (this is a slight abuse of notation, but it cannot cause confusion within formulas).
- We already gave an overview of the diagonalization of a formula during the informal proof of incompleteness. The idea is to consider  $\mathcal{G} = \mathcal{P}(\ulcorner \mathcal{G} \urcorner)$ .
- The goal of this chapter is to formalize the formula  $\mathcal{G}$  rigorously.

### 1.3. Primitive recursive functions

- Recall that primitive recursive functions are readily computable functions  $f : \mathbb{N}^p \rightarrow \mathbb{N}$ .
- The important point for this chapter is that the function  $\text{substitution}(n, v, k) : \mathbb{N}^3 \rightarrow \mathbb{N}$  is primitive recursive:  $n$  denotes the Gödel number of a formula  $\mathcal{P}$ ,  $v$  the code of a variable  $x$ , and  $k$  the value to be substituted for  $x$ . The function therefore returns  $\ulcorner \mathcal{P}(k) \urcorner \in \mathbb{N}$ .

## 1.4. Representable functions

- Every primitive recursive function is representable in  $\mathcal{PA}$ . We can therefore translate all our mathematical concepts into  $\mathcal{L}_0$ .
- For a primitive recursive function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , this means that there exists a formula  $\mathcal{P}(x, y)$  of  $\mathcal{L}_0$  such that, for every  $n \in \mathbb{N}$ , there is a proof in  $\mathcal{PA}$  of the equivalence:

$$\forall y \quad \mathcal{P}(\bar{n}, y) \leftrightarrow y = \overline{f(n)}$$

- The formula  $\mathcal{P}$  plays the role of the graph of  $f$ : it uniquely characterizes the value  $y = f(n)$  in  $\mathcal{L}_0$ .

## 2. Diagonal lemma

Let us tackle the cornerstone of the proof of the Incompleteness Theorem.

### 2.1. Statement

**Lemma 1** (Diagonal lemma).

For every formula  $\mathcal{P}(x)$  (where  $x$  is the only free variable), there exists a closed formula  $\mathcal{G}$  such that:

$$\boxed{\mathcal{PA} \vdash (\mathcal{G} \leftrightarrow \mathcal{P}(\ulcorner \mathcal{G} \urcorner))}$$

- Thus  $\mathcal{G}$  appears twice in the statement: once as a logical formula and once through its Gödel number. The formula  $\mathcal{G}$  therefore satisfies a property about itself.
- The diagonal lemma can be viewed as a fixed-point theorem: instead of obtaining  $f(x) = x$ , we have  $\mathcal{P}(\ulcorner \mathcal{G} \urcorner) \leftrightarrow \mathcal{G}$ .
- The equivalence is the syntactically correct way to say that  $\mathcal{G}$  and  $\mathcal{P}(\ulcorner \mathcal{G} \urcorner)$  are equivalent statements (but they are not equal formulas in  $\mathcal{L}_0$ ). On the other hand, the proof below will show that  $\mathcal{G}$  is defined exactly by  $\mathcal{G} = \mathcal{H}(\ulcorner \mathcal{H} \urcorner)$ .

### 2.2. Proof

#### Diagonalization on natural numbers

We will construct a primitive recursive function  $\text{diag} : \mathbb{N} \rightarrow \mathbb{N}$ .

- Given  $\mathcal{Q}(x)$ , we know how to assign it its Gödel number  $n = \ulcorner \mathcal{Q} \urcorner \in \mathbb{N}$ . We will consider the reverse operation.
- For  $n \in \mathbb{N}$ , we can decide whether this natural number corresponds to a well-formed formula  $\mathcal{Q}(x)$  of  $\mathcal{L}_0$  whose only free variable is  $x$ . If it does not, set  $\text{diag}(n) = 0$ .
- If  $n$  represents the formula  $\mathcal{Q}(x)$ , then we can evaluate  $\mathcal{Q}$  at  $n$ , that is, form the substitution  $\mathcal{Q}(\bar{n})$ . This is a formula of  $\mathcal{L}_0$  whose Gödel number is  $\ulcorner \mathcal{Q}(\bar{n}) \urcorner$ . We then define the natural number:

$$\text{diag}(n) = \ulcorner \mathcal{Q}(\bar{n}) \urcorner$$

That is,  $\text{diag}(n) = \ulcorner \mathcal{Q}(\ulcorner \mathcal{Q} \urcorner) \urcorner$ .

- Let us summarize the construction with a diagram:

$$n \in \mathbb{N} \xrightarrow{\text{Gödel decoding}} Q \in \mathcal{L}_0 \xrightarrow{\text{substitution}} Q(\bar{n}) \in \mathcal{L}_0 \xrightarrow{\text{Gödel number}} \text{diag}(n) = \ulcorner Q(\bar{n}) \urcorner \in \mathbb{N}$$

- For a formula  $Q(x)$  with Gödel number  $n = \ulcorner Q \urcorner$ , the computation of  $Q(k)$  is performed by the function  $\text{substitution}(n, \ulcorner x \urcorner, k)$ . For the formula  $Q$  with Gödel number  $n$ , this function replaces the occurrences of the variable  $x$  by  $\bar{k}$  and then returns the Gödel number of the resulting formula, namely  $\ulcorner Q(\bar{k}) \urcorner$ .

Remark: throughout the rest of the chapter, the variable being replaced will always be  $x$ .

- Deciding whether a natural number  $n$  denotes a well-formed formula  $Q(x)$  whose only free variable is  $x$  is a primitive recursive operation. Substitution is primitive recursive as well (see the chapters “Gödel coding” and “Primitive recursive functions”). Thus the function  $\text{diag} : \mathbb{N} \rightarrow \mathbb{N}$  is also primitive recursive.

### Diagonalization in the language

- Every primitive recursive function is representable (see the chapter “Representable Functions”). Therefore, for our function  $\text{diag} : \mathbb{N} \rightarrow \mathbb{N}$ , there exists a formula  $\text{Diag}(x, y)$  of  $\mathcal{L}_0$  such that, for every  $n \in \mathbb{N}$ :

$$\mathcal{PA} \vdash \left( \forall y \text{ Diag}(\bar{n}, y) \leftrightarrow y = \overline{\text{diag}(n)} \right)$$

### The formula $\mathcal{G}$

- Fix the formula  $\mathcal{P}(x)$  from the statement and define a new formula  $\mathcal{H}$  by:

$$\mathcal{H}(x) : \exists y \text{ Diag}(x, y) \wedge \mathcal{P}(y)$$

Let  $p$  denote the Gödel number of this last formula:  $p = \ulcorner \mathcal{H} \urcorner$ .

- Define the Gödel formula by:

$$\mathcal{G} = \mathcal{H}(\bar{p})$$

Thus, by definition,  $\mathcal{G}$  denotes “ $\mathcal{H}(\bar{p})$ ” and therefore also “ $\exists y \text{ Diag}(\bar{p}, y) \wedge \mathcal{P}(y)$ ”.

- Moreover,  $\text{diag}(p) = \ulcorner \mathcal{G} \urcorner$ . Indeed, apply the diagram above to the natural number  $p$ :

$$p \in \mathbb{N} \xrightarrow{\text{Gödel decoding}} \mathcal{H}(x) \in \mathcal{L}_0 \xrightarrow{\text{substitution}} \mathcal{H}(\bar{p}) \in \mathcal{L}_0 \xrightarrow{\text{Gödel number}} \text{diag}(p) = \ulcorner \mathcal{H}(\bar{p}) \urcorner = \ulcorner \mathcal{G} \urcorner \in \mathbb{N}$$

### End of the proof

Recall that the goal is to show:

$$\mathcal{PA} \vdash \left( \mathcal{G} \leftrightarrow \mathcal{P}(\ulcorner \mathcal{G} \urcorner) \right)$$

We therefore work in the theory  $\mathcal{PA}$ . Recall that  $p = \ulcorner \mathcal{H} \urcorner$ ,  $\mathcal{G} = \mathcal{H}(\bar{p})$ , and  $\text{diag}(p) = \ulcorner \mathcal{G} \urcorner$ .

- **Forward direction**  $\rightarrow$

Suppose  $\mathcal{G}$

therefore  $\mathcal{H}(\bar{p})$  by the definition of  $\mathcal{G}$

therefore  $\exists y \text{ Diag}(\bar{p}, y) \wedge \mathcal{P}(y)$  by the definition of  $\mathcal{H}$

therefore  $\exists y (y = \overline{\text{diag}(p)}) \wedge (\mathcal{P}(y))$  since  $\text{diag}$  is represented by  $\text{Diag}$  in  $\mathcal{PA}$

therefore  $\mathcal{P}(\overline{\text{diag}(p)})$  because the equality determines the value of  $y$

therefore  $\mathcal{P}(\ulcorner \mathcal{G} \urcorner)$

Thus  $\mathcal{G} \rightarrow \mathcal{P}(\ulcorner \mathcal{G} \urcorner)$ .

- **Reverse direction**  $\leftarrow$

Suppose  $\mathcal{P}(\ulcorner \mathcal{G} \urcorner)$ . We must deduce  $\mathcal{G} = \mathcal{H}(\overline{p})$ , that is:

$$\exists y \quad \text{Diag}(\overline{p}, y) \wedge \mathcal{P}(y)$$

We will show that  $y = \ulcorner \mathcal{G} \urcorner$  works.

- On the one hand,  $y = \ulcorner \mathcal{G} \urcorner = \overline{\text{diag}(\overline{p})}$ , so  $\text{Diag}(\overline{p}, y)$  holds by representability.
- Moreover,  $\mathcal{P}(y) = \mathcal{P}(\ulcorner \mathcal{G} \urcorner)$  is our assumption.
- Therefore, for this  $y$ , we do have  $\text{Diag}(\overline{p}, y) \wedge \mathcal{P}(y)$ .

By  $\exists$  introduction, we have:

$$\mathcal{P}(\ulcorner \mathcal{G} \urcorner) \rightarrow [\exists y \quad \text{Diag}(\overline{p}, y) \wedge \mathcal{P}(y)]$$

which is exactly  $\mathcal{P}(\ulcorner \mathcal{G} \urcorner) \rightarrow \mathcal{G}$ .

This completes the proof of the diagonal lemma. We have shown:

$$\mathcal{P}\mathcal{A} \vdash (\mathcal{G} \leftrightarrow \mathcal{P}(\ulcorner \mathcal{G} \urcorner))$$

### 3. Tarski's theorem

#### 3.1. Motivation

The rest of this chapter is not needed for the proof of the Incompleteness Theorem. However, it is an excellent application of the diagonal lemma and another opportunity to emphasize the difference between truth and provability. Tarski's theorem is a powerful version of the liar paradox. I make the following statement:

“I am lying right now.”

- If the statement is true, then I am lying, so the statement is false. Contradiction.
- If the statement is false, then I am not lying, so the statement is true. Another contradiction.

A variant is:

“This statement is false.”

#### 3.2. Truth is not definable

Informally, Tarski's theorem (1933) says that there is no formula  $\mathcal{V}(x)$  that recognizes true formulas; that is, there is no formula  $\mathcal{V}(x)$  such that  $\mathcal{V}(\overline{n})$  is true exactly when the closed formula  $\mathcal{P}$  with Gödel number  $n$  is true.

In our context, a closed formula  $\mathcal{P}$  is true if  $\models_{\mathbb{N}} \mathcal{P}$ , that is, if it is satisfied in the standard model of the natural numbers  $\mathbb{N}$ .

Here is the precise statement.

**Theorem 1** (Tarski).

*There is no formula  $\mathcal{V}(x)$  of  $\mathcal{L}_0$  such that, for every closed formula  $\mathcal{P}$  of  $\mathcal{L}_0$ :*

$$\models_{\mathbb{N}} \mathcal{P} \quad \text{if and only if} \quad \models_{\mathbb{N}} \mathcal{V}(\ulcorner \mathcal{P} \urcorner)$$

An equivalent formulation is “truth is not definable”. Indeed, let  $A$  be the set of natural numbers  $n = \ulcorner \mathcal{P} \urcorner$  such that  $\models_{\mathbb{N}} \mathcal{P}$  (that is, the Gödel numbers of true formulas). Then there is no formula  $\mathcal{V}(x)$  such that  $\models_{\mathbb{N}} \mathcal{V}(\overline{n})$  holds for every  $n \in A$  and only for those  $n$ . For this notion of “definable”, see the subsection “Definable sets” in the chapter “Peano arithmetic”. On the other hand, the next chapter will show that “being provable” is definable.

### 3.3. Proof

Suppose for a contradiction that such a formula  $\mathcal{V}(x)$  exists. Apply the diagonal lemma to  $\neg\mathcal{V}(x)$  (in place of  $\mathcal{P}(x)$  in the statement of the lemma). Thus there exists a closed formula  $\mathcal{G}$  such that:

$$\mathcal{P}\mathcal{A} \vdash (\mathcal{G} \leftrightarrow \neg\mathcal{V}(\overline{\mathcal{G}}))$$

This sentence  $\mathcal{G}$  is our version of the liar talking about itself. We work in the structure  $\mathbb{N}$  for the theory  $\mathcal{P}\mathcal{A}$  over  $\mathcal{L}_0$ . By the Soundness Theorem, if  $\mathcal{P}\mathcal{A} \vdash \mathcal{Q}$ , then  $\models_{\mathbb{N}} \mathcal{Q}$ : everything provable is true. Thus the equivalence above is true:

$$\mathcal{G} \leftrightarrow \neg\mathcal{V}(\overline{\mathcal{G}})$$

Let us verify that the sentence  $\mathcal{G}$  contradicts the equivalence required in Theorem 1 (where  $\mathcal{G}$  plays the role of the sentence  $\mathcal{P}$ ).

- If  $\mathcal{G}$  is true, that is,  $\models_{\mathbb{N}} \mathcal{G}$ , then by the equivalence from the diagonal lemma we also have  $\models_{\mathbb{N}} \neg\mathcal{V}(\overline{\mathcal{G}})$ . In other words,  $\mathcal{V}(\overline{\mathcal{G}})$  is false, so we do not have  $\models_{\mathbb{N}} \mathcal{V}(\overline{\mathcal{G}})$ . Thus, in this case,  $\mathcal{V}$  fails to recognize that  $\mathcal{G}$  is true.
- If  $\mathcal{G}$  is false (that is, we do not have  $\models_{\mathbb{N}} \mathcal{G}$ ), then by the equivalence from the diagonal lemma,  $\neg\mathcal{V}(\overline{\mathcal{G}})$  is false, so  $\mathcal{V}(\overline{\mathcal{G}})$  is true. Thus we have  $\models_{\mathbb{N}} \mathcal{V}(\overline{\mathcal{G}})$ . Consequently, in this case,  $\mathcal{V}$  fails to recognize that  $\mathcal{G}$  is false.
- Thus, without knowing whether  $\mathcal{G}$  is true or false,  $\mathcal{G}$  is nevertheless a counterexample to the desired equivalence.

# Proof of the Incompleteness Theorem

## Chapter 18

We bring together all the tools developed so far to prove Gödel's First Incompleteness Theorem.

Note: This is a good time to reread the chapter “An Overview of Gödel's Incompleteness Theorem” and also the preceding chapter, “Diagonalization.”

## 1. The incompleteness of Peano arithmetic

### 1.1. Statement

#### Theorem 1.

If  $\mathcal{PA}$  is consistent, then  $\mathcal{PA}$  is incomplete; that is, there exists a closed formula  $\mathcal{G}$  such that:

$$\mathcal{PA} \not\vdash \mathcal{G} \quad \text{and} \quad \mathcal{PA} \not\vdash \neg\mathcal{G}$$

Thus there exist statements that are true (in the standard model  $\mathbb{N}$ ) but unprovable.

- Recall that  $\mathcal{PA}$  denotes the theory of Peano arithmetic, that is, the usual axioms associated with the natural numbers.
- Incompleteness means that there exists a closed formula  $\mathcal{G}$  that is not provable from the Peano axioms and whose negation  $\neg\mathcal{G}$  is not provable either.
- Since  $\mathcal{G}$  is a closed formula, in every interpretation—for example, the model  $\mathbb{N}$  or another model—exactly one of the following two possibilities holds: either  $\mathcal{G}$  is true or  $\neg\mathcal{G}$  is true. One of them is true, and both are unprovable.
- Thus, for the standard model  $\mathbb{N}$ , and for every other model, there exist statements that are true (in that model) and unprovable (in  $\mathcal{PA}$ ).

### 1.2. Consistency and incompleteness

#### Review of consistency.

- A theory  $\mathcal{T}$  is **inconsistent** if there exists a formula  $\mathcal{Q}$  such that:

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{and} \quad \mathcal{T} \vdash \neg\mathcal{Q}.$$

It is called **consistent** otherwise.

- Equivalently,  $\mathcal{T}$  is inconsistent if  $\mathcal{T} \vdash \perp$ , in which case  $\mathcal{T}$  proves every formula (and its negation) by the rule of explosion. An inconsistent theory is therefore of no interest.

- If  $\mathcal{T}$  has a model (that is,  $\models_S \mathcal{T}$  for some structure  $S$ ), then  $\mathcal{T}$  is consistent by the Soundness Theorem.

### Review of completeness.

- A theory  $\mathcal{T}$  is **incomplete** if there exists a closed formula  $\mathcal{Q}$  such that:

$$\mathcal{T} \not\vdash \mathcal{Q} \quad \text{and} \quad \mathcal{T} \not\vdash \neg \mathcal{Q}.$$

- It is called **complete** otherwise; that is, for every closed formula  $\mathcal{Q}$ , we have:

$$\mathcal{T} \vdash \mathcal{Q} \quad \text{or} \quad \mathcal{T} \vdash \neg \mathcal{Q}.$$

In the statement of Gödel's theorem, we start with a consistent theory and prove that it is incomplete.

## 1.3. The consistency assumption

When we stated the Incompleteness Theorem in the chapter “An Overview of Gödel's Incompleteness Theorem,” we did not state the assumption “ $\mathcal{PA}$  is consistent.” Why not? This is one of the subtleties in the statement.

If we work in a world in which the natural numbers  $\mathbb{N}$  exist (for example, in  $\mathcal{ZFC}$ ), then  $\mathbb{N}$  is a model of  $\mathcal{PA}$  (everything is designed that way: the axioms of  $\mathcal{PA}$  correspond to properties of the natural numbers), so  $\mathcal{PA}$  is consistent, making the assumption unnecessary.

But if we wish to remain strictly within the framework of the language  $\mathcal{L}_0$  and the theory  $\mathcal{PA}$ , then the existence of  $\mathbb{N}$  is not given, and consequently neither is the consistency of  $\mathcal{PA}$ .

This issue is in fact unavoidable: the Second Incompleteness Theorem will state that a (sufficiently strong) theory cannot prove its own consistency.

## 2. $\omega$ -consistency, being a proof, and being provable

### 2.1. $\omega$ -consistency

We will prove Gödel's theorem under a slightly stronger assumption:  $\omega$ -consistency (the assumption used in Gödel's original article). A little later, we will discuss how to dispense with it.

#### Definition.

A theory  $\mathcal{T}$  is  **$\omega$ -inconsistent** if there exists a formula  $\mathcal{Q}(x)$  (with  $x$  as its only free variable) such that:

$$\mathcal{T} \vdash \exists x \mathcal{Q}(x)$$

$$\text{and for every } n \in \mathbb{N}, \quad \mathcal{T} \vdash \neg \mathcal{Q}(\bar{n})$$

Otherwise it is called  **$\omega$ -consistent**.

- This is therefore a type of inconsistency specific to the natural numbers: we have  $\neg \mathcal{Q}(0)$ ,  $\neg \mathcal{Q}(1)$ ,  $\neg \mathcal{Q}(2)$ , and so on, and yet there exists an  $x$  such that  $\mathcal{Q}(x)$ .
- In an  $\omega$ -consistent theory, no such formula exists.
- This definition goes beyond the language  $\mathcal{L}_0$  because it involves the natural numbers  $n = 0$ ,  $n = 1$ , ...
- If  $\mathcal{T}$  has  $\mathbb{N}$  as a model, then  $\mathcal{T}$  is  $\omega$ -consistent. Hence, if we regard  $\mathbb{N}$  as a model of  $\mathcal{PA}$ , then  $\mathcal{PA}$  is  $\omega$ -consistent.

Here is the connection with ordinary consistency:

- If  $\mathcal{T}$  is  $\omega$ -consistent, then  $\mathcal{T}$  is consistent.
- If  $\mathcal{T}$  is inconsistent, then  $\mathcal{T}$  is  $\omega$ -inconsistent.

Justification: if  $\mathcal{T}$  is inconsistent, then  $\mathcal{T}$  proves everything, and therefore proves the formulas in the definition of  $\omega$ -inconsistency. The other direction follows by contraposition.

These two notions are not equivalent, but a little later we will see Rosser's trick, which makes it possible to replace Gödel's assumption of " $\omega$ -consistency" with the weaker assumption of "consistency."

## 2.2. Being a proof and being provable

We return to the fundamental distinction between checking that something is a proof and deciding whether a formula is provable.

- Recall that  $\text{is\_proof}(N, n)$ , which decides whether the sequence of formulas with super-Gödel number  $N$  constitutes a proof of the formula whose Gödel number is  $n$ , is a primitive recursive predicate.
- Therefore, by representability, there is a corresponding formula in  $\mathcal{L}_0$ :

$$\mathcal{I}\text{s-proof}(y, x)$$

- On the other hand,  $\text{is\_provable}(n)$  is not a primitive recursive predicate, because the search for a proof must range over an unbounded set of integers  $N$ .
- But this is not a problem, because in  $\mathcal{L}_0$  it is perfectly legitimate to define the formula:

$$\mathcal{I}\text{s-provable}(x) : \exists y \mathcal{I}\text{s-proof}(y, x)$$

Note that to obtain this formula of  $\mathcal{L}_0$ , we used the representability of the relation "being a proof" as an intermediate step.

### Corollary 1.

*The set of provable formulas is definable.*

Indeed, by definition, a formula  $\mathcal{P}$  is provable (in  $\mathcal{PA}$ ) if and only if  $\mathcal{I}\text{s-provable}(\ulcorner \mathcal{P} \urcorner)$  is true (in  $\mathbb{N}$ ).

This stands in contrast to the property "being true," which is not definable (see Tarski's theorem in the chapter "Diagonalization").

Remember these two crucial differences:

- "Being a proof" and "being provable" are fundamentally different predicates: one is easy to compute, while the other is not.
- "Being provable" and "being true" are also fundamentally different concepts: one is definable, while the other is not.

### 3. Proof of the Incompleteness Theorem for $\mathcal{PA}$

Here is the statement that we will prove:

**Theorem 2.**

If  $\mathcal{PA}$  is  $\omega$ -consistent, then  $\mathcal{PA}$  is incomplete.

#### 3.1. The Gödel formula

Recall that the formula  $\text{Is-proof}(y, x)$  represents in  $\mathcal{PA}$  the predicate “ $y$  is the code of a proof of the formula with code  $x$ .” We then define:

$$\text{Is-provable}(x) : \exists y \text{ Is-proof}(y, x)$$

Consider the formula

$$\mathcal{P}(x) : \neg \text{Is-provable}(x)$$

By the diagonal lemma, there exists a closed formula  $\mathcal{G}$ , called the *Gödel formula*, such that

$$\mathcal{PA} \vdash (\mathcal{G} \leftrightarrow \mathcal{P}(\ulcorner \mathcal{G} \urcorner))$$

(We use  $\ulcorner \mathcal{G} \urcorner$  both for the Gödel number of  $\mathcal{G}$  and for its numeral.)

Thus, in  $\mathcal{PA}$ , the formula  $\mathcal{G}$  states that it is not provable. In other words, it says: “I am not provable.” We will now show that this formula  $\mathcal{G}$  is not provable and that  $\neg \mathcal{G}$  is not provable either.

#### 3.2. Proof that $\mathcal{PA} \not\vdash \mathcal{G}$

Suppose for a contradiction that  $\mathcal{PA} \vdash \mathcal{G}$ . Then there exists a proof of  $\mathcal{G}$  from the axioms of  $\mathcal{PA}$ . Let  $N$  denote the super-Gödel number associated with the formulas that constitute this proof.

Thus  $\text{is\_proof}(N, \ulcorner \mathcal{G} \urcorner)$  holds (as a primitive recursive predicate), and therefore:

$$\begin{aligned} \mathcal{PA} &\vdash \text{Is-proof}(\overline{N}, \ulcorner \mathcal{G} \urcorner) \\ \text{therefore } \mathcal{PA} &\vdash \exists y \text{ Is-proof}(y, \ulcorner \mathcal{G} \urcorner) \\ \text{therefore } \mathcal{PA} &\vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \\ \text{therefore } \mathcal{PA} &\vdash \neg \mathcal{G} \text{ by the diagonal equivalence.} \end{aligned}$$

Conclusion: if  $\mathcal{PA} \vdash \mathcal{G}$ , then  $\mathcal{PA} \vdash \neg \mathcal{G}$ , which implies that  $\mathcal{PA}$  is inconsistent. But we assumed that  $\mathcal{PA}$  is  $\omega$ -consistent and hence consistent (this weaker assumption is enough for this first step). We obtain a contradiction.

#### 3.3. Proof that $\mathcal{PA} \not\vdash \neg \mathcal{G}$

Suppose for a contradiction that  $\mathcal{PA} \vdash \neg \mathcal{G}$ . Then, by the diagonal equivalence, we have:

$$\mathcal{PA} \vdash \exists y \text{ Is-proof}(y, \ulcorner \mathcal{G} \urcorner)$$

If we define  $Q(y)$  by  $\text{Is-proof}(y, \ulcorner \mathcal{G} \urcorner)$ , this can be written:

$$\mathcal{PA} \vdash \exists y Q(y)$$

On the other hand, since we assume  $\mathcal{PA} \vdash \neg \mathcal{G}$  and  $\mathcal{PA}$  is consistent (because it is  $\omega$ -consistent), we have  $\mathcal{PA} \not\vdash \mathcal{G}$  (which we in fact already proved in the preceding section).

This means that for every integer  $N = 0, 1, 2, \dots$ , it is false that  $N$  is a proof of  $\ulcorner \mathcal{G} \urcorner$ ; in other words,  $\neg \text{is\_proof}(N, \ulcorner \mathcal{G} \urcorner)$  is true. By the (strong) representability of primitive recursive predicates, we have:

$$\mathcal{PA} \vdash \neg \text{Is-proof}(\overline{N}, \ulcorner \mathcal{G} \urcorner)$$

This means that for  $N = 0, 1, 2, \dots$ , we have

$$\mathcal{PA} \vdash \neg Q(\bar{N})$$

This implies that  $\mathcal{PA}$  is  $\omega$ -inconsistent, contradicting our assumption.

### 3.4. Is $\mathcal{G}$ true?

We have proved  $\mathcal{PA} \not\vdash \mathcal{G}$  and  $\mathcal{PA} \not\vdash \neg\mathcal{G}$ . Suppose we work in a model of  $\mathcal{PA}$ . Then:

- either the sentence  $\mathcal{G}$  is true and is not provable,
- or the sentence  $\neg\mathcal{G}$  is true and is not provable either.

In either case, there are unprovable truths!

Let us work in the usual setting in which  $\mathbb{N}$  is regarded as a model of  $\mathcal{PA}$  (for example, within  $\mathcal{ZFC}$ ). On the one hand,  $\mathcal{PA}$  is  $\omega$ -consistent (and therefore consistent). On the other hand,  $\mathcal{G}$  is true! Indeed,  $\mathcal{G}$  is not provable, which means that the formula  $\neg\text{Is-provable}(\ulcorner \mathcal{G} \urcorner)$  is true (in  $\mathbb{N}$ ). Therefore, by the diagonal equivalence,  $\mathcal{G}$  is true.

Once again, this assumes that  $\mathbb{N}$  is a model of  $\mathcal{PA}$  (an assumption that lies outside the framework of Peano arithmetic). This is why Gödel states his theorem with a consistency assumption and without a notion of truth, so as to remain strictly within the framework of  $\mathcal{PA}$ .

### 3.5. Rosser's trick

Rosser (1936) replaces Gödel's assumption of  $\omega$ -consistency with the weaker assumption " $\mathcal{PA}$  is consistent." Instead of using the sentence "I am not provable" in the proof, he uses the sentence: "If I am provable, then there exists a shorter proof of my negation."

Formally, the formula  $\mathcal{P}(x)$  to which he applies the diagonal lemma is:

$$\mathcal{P}(x) : \quad \forall y \left( \text{Is-proof}(y, x) \rightarrow \exists z \leq y \text{ Is-proof}(z, \text{neg}(x)) \right)$$

I.e., it says: "If I am provable, then there exists a proof of my negation whose code is no greater than the code of my proof."

If  $x = \ulcorner \mathcal{Q} \urcorner$ , we write  $\text{neg}(x) = \ulcorner \neg\mathcal{Q} \urcorner$ .

Notice the bounded quantifier " $\exists z \leq y$ ", which avoids the need to introduce  $\omega$ -consistency and enumerate  $N = 0, 1, 2, \dots$  all the way to infinity.

## 4. The general Incompleteness Theorem

### 4.1. Statement

#### Theorem 3.

Let  $\mathcal{T}$  be a consistent theory with recursive axioms that contains  $\mathcal{PA}$ . Then  $\mathcal{T}$  is incomplete.

In other words, in any sufficiently strong theory, there are true but unprovable formulas.

Since Peano arithmetic is incomplete, one might think that it would be enough to add an unprovable formula (such as  $\mathcal{G}$ ) to its axioms in order to make it immediately provable in this new theory. But Gödel's theorem proves that this new theory will also be incomplete: there will be a new unprovable formula. There is therefore no way out: there will always be true but unprovable formulas.

Let us revisit the assumptions. We have already recalled what it means for " $\mathcal{T}$  to be consistent." The assumption " $\mathcal{T}$  contains  $\mathcal{PA}$ " means exactly " $\mathcal{PA} \subset \mathcal{T}$ "; that is, the Peano axioms are among the axioms of  $\mathcal{T}$ . Informally, we say that  $\mathcal{T}$  is "sufficiently strong." Being able to perform computations with integers is, however, a fairly weak requirement in mathematics.

It remains to explain the assumption “the axioms of  $\mathcal{T}$  are recursive.”

## 4.2. Recursive functions: Church’s thesis

We will explain, without giving a complete definition, the notion of “recursive,” which is broader than “primitive recursive.”

**Informal definition.** A function  $f : \mathbb{N}^p \rightarrow \mathbb{N}$  is a **recursive function** if it is computable.

The class of recursive functions is therefore larger than the class of primitive recursive functions, which were the “easily computable” functions.

There is, of course, a precise definition of a recursive function. It is a more technical variant of the notion of a primitive recursive function, with unbounded operations added. For example, the Ackermann function (see the chapter “Primitive Recursive Functions”) is recursive but not primitive recursive.

**Church’s thesis** is the heuristic claim that all computable functions are recursive (and conversely).

The result that matters to us is that every recursive function is representable (as with primitive recursive functions).

Let us finish our definitions:

- A set  $A \subset \mathbb{N}$  is a **recursive set** (respectively, a primitive recursive set) if its characteristic function  $\chi_A : \mathbb{N} \rightarrow \mathbb{N}$  is recursive (respectively, primitive recursive).
- A theory  $\mathcal{T}$  has **recursive axioms** (respectively, primitive recursive axioms) if  $A = \{\ulcorner \mathcal{P} \urcorner \mid \mathcal{P} \text{ is an axiom of } \mathcal{T}\}$ , the set of Gödel numbers of the axioms of  $\mathcal{T}$ , is recursive (respectively, primitive recursive).

Requiring the set of axioms of a theory to be recursive simply means that we can determine whether or not a given formula is an axiom. This is an extremely reasonable requirement: we must be able to recognize the axioms of our theory!

## 4.3. Proof of the Incompleteness Theorem

Since we have not given the exact definition of a recursive function, we replace this assumption with the slightly stronger assumption “the axioms of  $\mathcal{T}$  are primitive recursive,” which means that we can (easily) determine whether or not a given formula is an axiom.

This makes it possible to construct the primitive recursive predicate  $\text{is\_proof}(N, n) : \mathbb{N}^2 \rightarrow \{T, F\}$ , and then represent it by a formula of  $\mathcal{L}_0$ ,  $\mathcal{I}s\text{-proof}(y, x)$ . We also replace the consistency assumption with the stronger assumption of  $\omega$ -consistency.

The proof is then exactly the same as for the theory  $\mathcal{PA}$ : we construct a formula  $\mathcal{G}$  by diagonalization and prove that neither  $\mathcal{G}$  nor  $\neg\mathcal{G}$  is provable.

If  $\mathbb{N}$  is a model of the theory  $\mathcal{T}$ , then  $\mathcal{G}$  is true, although unprovable.

### Returning to completeness.

By the Completeness Theorem, we know that if a formula is true in every model, then it is provable. This does not contradict the Incompleteness Theorem at all; it simply means here that there are models in which  $\mathcal{G}$  is false. These models are called *nonstandard models* of  $\mathcal{PA}$ : they satisfy the Peano axioms but are not equal to  $\mathbb{N}$  (for example, they contain infinitely large integers).

# The Second Incompleteness Theorem

We conclude this book with Gödel's Second Incompleteness Theorem.

## 1. Gödel's Second Theorem

### 1.1. Statement

#### Theorem 1.

Let  $\mathcal{T}$  be a consistent theory containing  $\mathcal{PA}$  and having recursive axioms. Then  $\mathcal{T}$  cannot prove its own consistency.

For a fixed theory  $\mathcal{T}$ , let  $\text{Is-consistent}(\mathcal{T})$  denote the sentence expressing that  $\mathcal{T}$  is consistent. The conclusion of the theorem is therefore:

$$\mathcal{T} \not\vdash \text{Is-consistent}(\mathcal{T})$$

Notice that the assumptions are the same as those of the First Incompleteness Theorem. In fact, we will prove the Second Incompleteness Theorem from the first, assuming its formalized version.

### 1.2. Consistency

Recall that a theory is *inconsistent* if it proves every formula, in particular the absurd formula " $0 = 1$ ". This can be expressed in the language  $\mathcal{L}_0$  by  $\text{Is-provable}(\ulcorner 0 = 1 \urcorner)$ , where of course we write  $1 = S(0)$ . We therefore define the formula:

$$\text{Is-consistent}(\mathcal{T}) : \neg \text{Is-provable}(\ulcorner 0 = 1 \urcorner)$$

Thus the consistency of  $\mathcal{T}$  can be expressed by a sentence of  $\mathcal{L}_0$ .

## 2. Proof

The proof follows fairly directly from the First Incompleteness Theorem if we accept its formalized version.

### 2.1. Formalization of the First Incompleteness Theorem

Suppose that  $\mathcal{T}$  is a theory containing  $\mathcal{PA}$  and that the axioms of  $\mathcal{T}$  are recursive. We focus only on the consistency assumption.

Recall that the diagonal lemma provides a formula  $\mathcal{G}$  such that:

$$\mathcal{T} \vdash (\mathcal{G} \leftrightarrow \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner))$$

The statement of the First Incompleteness Theorem is then:

If  $\mathcal{T}$  is consistent, then  $\mathcal{G}$  is unprovable.

But by the diagonal lemma,  $\mathcal{G}$  means exactly “ $\mathcal{G}$  is unprovable,” so the statement of the first theorem is also:

If  $\mathcal{T}$  is consistent, then  $\mathcal{G}$ .

Here is the formalization of the First Incompleteness Theorem:

$$\mathcal{T} \vdash (\text{Is-consistent}(\mathcal{T}) \rightarrow \mathcal{G})$$

This is exactly the preceding statement, now *internalized*. The statement “if  $\mathcal{T}$  is consistent, then  $\mathcal{G}$ ” is provable (as we saw in the preceding chapters), but here we assume that a proof of it could be written within the theory  $\mathcal{T}$ . We will return to this formalization in the last section.

## 2.2. Proof of the Second Incompleteness Theorem

We assume the formalized version of the First Incompleteness Theorem in order to prove the second. In doing so, we follow the same approach as Gödel when he states the second theorem and sketches its proof at the end of his foundational paper on the Incompleteness Theorems.

We also need to assume the principle called “proof necessitation”:

$$\text{if } \mathcal{T} \vdash \mathcal{Q} \text{ then } \mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{Q} \urcorner)$$

This means that if a theory proves a formula, then it knows that it can prove that formula!

Let us now prove that a consistent theory cannot prove its own consistency.

We start with an arbitrary theory  $\mathcal{T}$  and suppose that it proves its own consistency:

$$\mathcal{T} \vdash \text{Is-consistent}(\mathcal{T})$$

By *modus ponens* applied to the formalization of the First Incompleteness Theorem, we then have:

$$\mathcal{T} \vdash \mathcal{G}$$

therefore  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner)$  by proof necessitation

therefore  $\mathcal{T} \vdash \neg \mathcal{G}$  by the diagonal equivalence

In summary:

$$\mathcal{T} \vdash \mathcal{G} \quad \text{and} \quad \mathcal{T} \vdash \neg \mathcal{G}$$

Therefore  $\mathcal{T}$  is inconsistent.

Thus, by contraposition, we have proved the Second Incompleteness Theorem:

If  $\mathcal{T}$  is consistent, then  $\mathcal{T} \nvdash \text{Is-consistent}(\mathcal{T})$ .

## 2.3. Returning to the formalization

The proof explained above is Gödel's original sketch (1931). A detailed proof would come only with the work of Hilbert and Bernays (1934–1939), followed by Löb (1955). This proof rests on three conditions, called the “Hilbert–Bernays–Löb derivability conditions”:

- **C1** - *Proof necessitation*.

$$\text{If } \mathcal{T} \vdash Q, \text{ then } \mathcal{T} \vdash \text{Is-provable}(\ulcorner Q \urcorner)$$

- **C2** - *Modus ponens for proofs*.

$$\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{P} \rightarrow Q \urcorner) \rightarrow (\text{Is-provable}(\ulcorner \mathcal{P} \urcorner) \rightarrow \text{Is-provable}(\ulcorner Q \urcorner))$$

- **C3** - *Proof introspection*.

$$\mathcal{T} \vdash \text{Is-provable}(\ulcorner Q \urcorner) \rightarrow \text{Is-provable}(\ulcorner \text{Is-provable}(\ulcorner Q \urcorner) \urcorner)$$

We assume that these conditions hold. (They are highly technical to prove; it would take until 1955 to complete their formalization.) We can now give a rigorous proof of the formalization of the First Incompleteness Theorem. We start from the equivalence provided by diagonalization:

$$\mathcal{T} \vdash \mathcal{G} \leftrightarrow \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner)$$

therefore  $\mathcal{T} \vdash \mathcal{G} \rightarrow \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner)$  keeping only the forward implication

therefore  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \rightarrow \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \urcorner)$  by (C1)

therefore  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \text{Is-provable}(\ulcorner \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \urcorner)$  by (C2)

But by (C3), we always have:

$$\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \text{Is-provable}(\ulcorner \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \urcorner)$$

Setting

$$Q : \text{Is-provable}(\ulcorner \mathcal{G} \urcorner),$$

we obtain two conclusions:

$$\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \text{Is-provable}(\ulcorner Q \urcorner) \wedge \text{Is-provable}(\ulcorner \neg Q \urcorner)$$

therefore  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \text{Is-provable}(\ulcorner Q \wedge \neg Q \urcorner)$

therefore  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \text{Is-provable}(\ulcorner 0 = 1 \urcorner)$

thus  $\mathcal{T} \vdash \text{Is-provable}(\ulcorner \mathcal{G} \urcorner) \rightarrow \neg \text{Is-consistent}(\mathcal{T})$

By contraposition:

$$\mathcal{T} \vdash \text{Is-consistent}(\mathcal{T}) \rightarrow \neg \text{Is-provable}(\ulcorner \mathcal{G} \urcorner)$$

But by diagonalization, the conclusion is exactly  $\mathcal{G}$ , and therefore:

$$\mathcal{T} \vdash \text{Is-consistent}(\mathcal{T}) \rightarrow \mathcal{G}$$

This is the desired formalization of the First Incompleteness Theorem.

# Appendix

## Gödel's life

Kurt Gödel was born in Brno in 1906 (then Austria-Hungary, now the Czech Republic) and died in Princeton (United States) in 1978. He began his studies in Vienna in 1923, where he became interested in physics, mathematics, and philosophy. He completed his thesis in 1929 under the supervision of Hans Hahn, working on problems posed by David Hilbert concerning the foundations of mathematics.

Hilbert's program sought to prove the consistency and completeness of mathematics. Gödel first responded with his Completeness Theorem: "What is true in every possible mathematical world is provable (and conversely)."

However, in 1930, he obtained his major result establishing the limits of mathematical theories: the Incompleteness Theorems. "For every theory rich enough to describe the world of the natural numbers, there are formulas that are true of these numbers but are not provable." The apparent contradiction with the preceding result is only superficial, because here we are interested solely in the specific world of the natural numbers. These results were published in the 1931 paper "On Formally Undecidable Propositions."

This work immediately attracted international attention and surprised and unsettled the mathematical community, beginning with Hilbert, whose credo was: "We must know, we will know." Hilbert was incredulous at first, but then acknowledged the validity of Gödel's proof and worked to disseminate its results. Indeed, the mathematical community had always thought that everything true ought to be provable. Gödel's theorems showed that this is impossible! This is the most important result in twentieth-century logic.

Gödel had opportunities to present his work around the world, particularly in the United States, where he met Albert Einstein as early as 1933. During the 1930s, Austria became fascist, culminating in its annexation by the Nazi regime (the Anschluss, March 1938). Gödel was not Jewish, but he had worked with many Jewish professors and was independently minded. This was enough for him to be sidelined at the university.

So, in December 1939, he left Vienna with his wife Adele for the United States. Rather than heading west across the Atlantic, they traveled east: they crossed Russia on the Trans-Siberian Railway, continued through Japan, crossed the Pacific to San Francisco, and arrived in Princeton (on the East Coast of the United States) in March 1940, where Einstein was waiting for him. All this through countries at war!

Here is another famous anecdote: when Gödel appeared before a judge to become a naturalized American citizen, Einstein served as his witness. For the occasion, Gödel had studied the U.S. Constitution, and before entering the courtroom he told Einstein that he had found a logical loophole in it. Fortunately, Einstein stopped Gödel from mentioning it to the judge!

Gödel and Einstein remained friends until Einstein's death in 1955. Every day, they walked and talked together on their way to Princeton. Gödel continued his work and turned to philosophy and the theory of relativity.

Throughout his life, Gödel was obsessed with his health. Increasingly hypochondriacal, he first feared accidental poisoning and then deliberate poisoning, becoming trapped in his paranoia. He saw almost no one except Einstein and his wife Adele (who, moreover, had to taste all his food). When his wife was hospitalized, he stopped eating and died in 1978 at the age of 71.

## Further reading

Here are some annotated references offering different perspectives and opportunities to explore the subject in greater depth.

- *forall x: An Introduction to Formal Logic*, by P. D. Magnus, T. Button, R. Trueman, and R. Zach.  
[forallx.openlogicproject.org](http://forallx.openlogicproject.org)

A good introduction to formal logic that covers true/false logic, first-order logic, and proofs in both systems.

The chapters are short, well motivated, and suitable for first- or second-year students who have never taken a course in logic (or even in mathematics).

The book includes a substantial amount of prose and examples drawn from everyday language. This makes it an accessible resource for students outside mathematics.

- *Mathematical Logic*, volumes 1 and 2, by R. Cori and D. Lascar, published by Oxford University Press.

These are reference works intended primarily for graduate students. The first volume covers propositional logic and first-order logic (and much more), while the second covers the Incompleteness Theorems (and much more). The work is comprehensive and very well written, and is suitable for readers who have already taken a logic course and have a solid mathematical background.

- *A Mathematical Introduction to Logic*, by H. Enderton, published by Academic Press. This is a standard reference for advanced undergraduate and graduate students. It covers propositional logic, first-order logic, the Incompleteness Theorems, and second-order logic.

- *Gödel Without (Too Many) Tears*, by P. Smith:

[logicmatters.net/igt](http://logicmatters.net/igt)

This book is devoted entirely to Gödel's Incompleteness Theorems. The author is a professor of logic and philosophy and adopts a more literary style aimed at a nonmathematical audience, with many conceptual explanations. This short version contains no proofs; they are included in the expanded version, *An Introduction to Gödel's Theorems*.

Tables

Connectives

| Symbol            | Name        | Definition                                    |
|-------------------|-------------|-----------------------------------------------|
| $\neg$            | NOT         | $\neg P$ is true iff $P$ is false             |
| $\wedge$          | AND         | $P \wedge Q$ is true iff $P$ and $Q$ are true |
| $\vee$            | OR          | $P \vee Q$ is true iff $P$ or $Q$ is true     |
| $\rightarrow$     | implication | $(\neg P) \vee Q$                             |
| $\leftrightarrow$ | equivalence | $(P \rightarrow Q) \wedge (Q \rightarrow P)$  |

Common logical equivalences in true/false logic

| Name                    | Rule                                                                                                                              |                                                                                                                                 |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| <b>Idempotence</b>      | $\mathcal{P} \wedge \mathcal{P} \equiv \mathcal{P}$                                                                               | $\mathcal{P} \vee \mathcal{P} \equiv \mathcal{P}$                                                                               |
| <b>Commutativity</b>    | $\mathcal{P} \wedge \mathcal{Q} \equiv \mathcal{Q} \wedge \mathcal{P}$                                                            | $\mathcal{P} \vee \mathcal{Q} \equiv \mathcal{Q} \vee \mathcal{P}$                                                              |
| <b>Associativity</b>    | $(\mathcal{P} \wedge \mathcal{Q}) \wedge \mathcal{R} \equiv \mathcal{P} \wedge (\mathcal{Q} \wedge \mathcal{R})$                  | $(\mathcal{P} \vee \mathcal{Q}) \vee \mathcal{R} \equiv \mathcal{P} \vee (\mathcal{Q} \vee \mathcal{R})$                        |
| <b>Distributivity</b>   | $\mathcal{P} \wedge (\mathcal{Q} \vee \mathcal{R}) \equiv (\mathcal{P} \wedge \mathcal{Q}) \vee (\mathcal{P} \wedge \mathcal{R})$ | $\mathcal{P} \vee (\mathcal{Q} \wedge \mathcal{R}) \equiv (\mathcal{P} \vee \mathcal{Q}) \wedge (\mathcal{P} \vee \mathcal{R})$ |
| <b>Absorption</b>       | $\mathcal{P} \wedge (\mathcal{P} \vee \mathcal{Q}) \equiv \mathcal{P}$                                                            | $\mathcal{P} \vee (\mathcal{P} \wedge \mathcal{Q}) \equiv \mathcal{P}$                                                          |
| <b>De Morgan's laws</b> | $\neg(\mathcal{P} \wedge \mathcal{Q}) \equiv (\neg \mathcal{P}) \vee (\neg \mathcal{Q})$                                          | $\neg(\mathcal{P} \vee \mathcal{Q}) \equiv (\neg \mathcal{P}) \wedge (\neg \mathcal{Q})$                                        |
| <b>Contrapositive</b>   | $\mathcal{P} \rightarrow \mathcal{Q} \equiv (\neg \mathcal{Q}) \rightarrow (\neg \mathcal{P})$                                    |                                                                                                                                 |

## Rules of inference for true/false logic

| Name                         | Rule                                                                                                                                                                                                                                                                                                      |
|------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Reiteration                  | $\mathcal{P} \vdash \mathcal{P}$                                                                                                                                                                                                                                                                          |
| $\wedge$ elimination         | $\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{P}$<br>$\mathcal{P} \wedge \mathcal{Q} \vdash \mathcal{Q}$                                                                                                                                                                                                |
| $\wedge$ introduction        | $\mathcal{P}, \mathcal{Q} \vdash \mathcal{P} \wedge \mathcal{Q}$                                                                                                                                                                                                                                          |
| Double-negation elimination  | $\neg\neg\mathcal{P} \vdash \mathcal{P}$                                                                                                                                                                                                                                                                  |
| $\vee$ introduction          | $\mathcal{P} \vdash \mathcal{P} \vee \mathcal{Q}$<br>$\mathcal{Q} \vdash \mathcal{P} \vee \mathcal{Q}$                                                                                                                                                                                                    |
| $\vee$ elimination           | $\mathcal{P} \vee \mathcal{Q}, \mathcal{P} \rightarrow \mathcal{R}, \mathcal{Q} \rightarrow \mathcal{R} \vdash \mathcal{R}$                                                                                                                                                                               |
| Implication (modus ponens)   | $\mathcal{P}, \mathcal{P} \rightarrow \mathcal{Q} \vdash \mathcal{Q}$                                                                                                                                                                                                                                     |
| $\rightarrow$ introduction   | If $\mathcal{P} \vdash \mathcal{Q}$ then $\vdash \mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                                                                                                                     |
| Explosion (ex falso)         | $\perp \vdash \mathcal{P}$                                                                                                                                                                                                                                                                                |
| Equivalence                  | $\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{P} \rightarrow \mathcal{Q}$<br>$\mathcal{P} \leftrightarrow \mathcal{Q} \vdash \mathcal{Q} \rightarrow \mathcal{P}$<br>$\mathcal{P} \rightarrow \mathcal{Q}, \mathcal{Q} \rightarrow \mathcal{P} \vdash \mathcal{P} \leftrightarrow \mathcal{Q}$ |
| Definition of negation       | $\neg\mathcal{P}$ is defined as $\mathcal{P} \rightarrow \perp$                                                                                                                                                                                                                                           |
| Double-negation introduction | $\mathcal{P} \vdash \neg\neg\mathcal{P}$                                                                                                                                                                                                                                                                  |
| Law of excluded middle       | $\vdash \mathcal{P} \vee \neg\mathcal{P}$                                                                                                                                                                                                                                                                 |
| Modus tollens                | $\mathcal{P} \rightarrow \mathcal{Q}, \neg\mathcal{Q} \vdash \neg\mathcal{P}$                                                                                                                                                                                                                             |
| De Morgan's laws             | $\neg(\mathcal{P} \wedge \mathcal{Q}) \vdash \neg\mathcal{P} \vee \neg\mathcal{Q}$<br>...                                                                                                                                                                                                                 |
| Proof by contradiction       | $\neg\mathcal{P} \rightarrow \perp \vdash \mathcal{P}$                                                                                                                                                                                                                                                    |
| Contrapositive               | $\mathcal{P} \rightarrow \mathcal{Q} \vdash \neg\mathcal{Q} \rightarrow \neg\mathcal{P}$<br>$\neg\mathcal{Q} \rightarrow \neg\mathcal{P} \vdash \mathcal{P} \rightarrow \mathcal{Q}$                                                                                                                      |

## Binary connectives

| Symbol            | Representation                       | Name                         | Values |
|-------------------|--------------------------------------|------------------------------|--------|
| $\perp$           | $P \wedge \neg P$                    | always false / contradiction | 0000   |
| $\wedge$          | $P \wedge Q$                         | AND/ conjunction             | 0001   |
|                   | $P \wedge \neg Q$                    | $P$ and (not $Q$ )           | 0010   |
|                   | $P$                                  | left projection              | 0011   |
|                   | $\neg P \wedge Q$                    | (not $P$ ) and $Q$           | 0100   |
|                   | $Q$                                  | right projection             | 0101   |
| $\oplus$          | $(P \vee Q) \wedge \neg(P \wedge Q)$ | XOR / exclusive or           | 0110   |
| $\vee$            | $P \vee Q$                           | OR/ disjunction              | 0111   |
| $\downarrow$      | $\neg(P \vee Q)$                     | NOR                          | 1000   |
| $\leftrightarrow$ | $P \leftrightarrow Q$                | equivalence                  | 1001   |
|                   | $\neg Q$                             | negation of $Q$              | 1010   |
|                   | $Q \rightarrow P$                    | $Q$ implies $P$              | 1011   |
|                   | $\neg P$                             | negation of $P$              | 1100   |
| $\rightarrow$     | $P \rightarrow Q$                    | $P$ implies $Q$              | 1101   |
| $\uparrow$        | $\neg(P \wedge Q)$                   | NAND / Sheffer               | 1110   |
| $\top$            | $P \vee \neg P$                      | always true / tautology      | 1111   |

The last column gives the 4 values of the function when  $(P, Q)$  is successively  $(0, 0)$ ,  $(0, 1)$ ,  $(1, 0)$ , and  $(1, 1)$ .

## Common logical equivalences in first-order logic

| Name                                    | Rule                                                                                                                                                        |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Negation of $\forall$                   | $\neg(\forall x \mathcal{P}(x)) \equiv \exists x \neg \mathcal{P}(x)$                                                                                       |
| Negation of $\exists$                   | $\neg(\exists x \mathcal{P}(x)) \equiv \forall x \neg \mathcal{P}(x)$                                                                                       |
| Permutation of $\forall$                | $\forall x \forall y \mathcal{P}(x, y) \equiv \forall y \forall x \mathcal{P}(x, y)$                                                                        |
| Permutation of $\exists$                | $\exists x \exists y \mathcal{P}(x, y) \equiv \exists y \exists x \mathcal{P}(x, y)$                                                                        |
| Distribution of $\forall$ over $\wedge$ | $\forall x (\mathcal{P}(x) \wedge \mathcal{Q}(x)) \equiv (\forall x \mathcal{P}(x)) \wedge (\forall x \mathcal{Q}(x))$                                      |
| Distribution of $\exists$ over $\vee$   | $\exists x (\mathcal{P}(x) \vee \mathcal{Q}(x)) \equiv (\exists x \mathcal{P}(x)) \vee (\exists x \mathcal{Q}(x))$                                          |
| Moving $\forall$ outward                | $\forall x (\mathcal{P}(x) \rightarrow \mathcal{Q}) \equiv (\exists x \mathcal{P}(x)) \rightarrow \mathcal{Q}$<br>(if $x$ does not occur in $\mathcal{Q}$ ) |
| Renaming a bound variable               | $\forall x \mathcal{P}(x) \equiv \forall y \mathcal{P}[y/x]$<br>(if $y$ is a new variable)                                                                  |

## Rules of inference for first-order logic

| Name                                     | Rule                                                                                                                                                                                                                                           |
|------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| $\forall$ elimination                    | $\forall x \mathcal{P}(x) \vdash \mathcal{P}(c)$<br>( $c$ is any constant or closed term)                                                                                                                                                      |
| $\forall$ introduction                   | $\mathcal{P}(c) \vdash \forall x \mathcal{P}(x)$<br>( $c$ is arbitrary: it does not occur in any open assumption)                                                                                                                              |
| $\exists$ introduction                   | $\mathcal{P}(c) \vdash \exists x \mathcal{P}(x)$<br>( $c$ is any constant or closed term)                                                                                                                                                      |
| $\exists$ elimination                    | $\exists x \mathcal{P}(x), [\mathcal{P}(c) \vdash \mathcal{Q}] \vdash \mathcal{Q}$<br>( $c$ is a fresh constant that does not occur anywhere else)                                                                                             |
| Negation of $\forall$                    | $\neg(\forall x \mathcal{P}(x)) \vdash \exists x \neg \mathcal{P}(x)$                                                                                                                                                                          |
| Negation of $\exists$                    | $\neg(\exists x \mathcal{P}(x)) \vdash \forall x \neg \mathcal{P}(x)$                                                                                                                                                                          |
| Renaming                                 | $\forall x \mathcal{P}(x) \vdash \forall y \mathcal{P}(y)$<br>$\exists x \mathcal{P}(x) \vdash \exists y \mathcal{P}(y)$                                                                                                                       |
| Permutation of two $\forall$ quantifiers | $\forall x \forall y \mathcal{P}(x, y) \vdash \forall y \forall x \mathcal{P}(x, y)$                                                                                                                                                           |
| Permutation of two $\exists$ quantifiers | $\exists x \exists y \mathcal{P}(x, y) \vdash \exists y \exists x \mathcal{P}(x, y)$                                                                                                                                                           |
| Interchanging $\exists$ and $\forall$    | $\exists x \forall y \mathcal{P}(x, y) \vdash \forall y \exists x \mathcal{P}(x, y)$                                                                                                                                                           |
| Commutation of $\forall$ and $\wedge$    | $\forall x(\mathcal{P}(x) \wedge \mathcal{Q}(x)) \vdash (\forall x \mathcal{P}(x)) \wedge (\forall x \mathcal{Q}(x))$<br>$(\forall x \mathcal{P}(x)) \wedge (\forall x \mathcal{Q}(x)) \vdash \forall x(\mathcal{P}(x) \wedge \mathcal{Q}(x))$ |
| Commutation of $\exists$ and $\vee$      | $\exists x(\mathcal{P}(x) \vee \mathcal{Q}(x)) \vdash (\exists x \mathcal{P}(x)) \vee (\exists x \mathcal{Q}(x))$<br>$(\exists x \mathcal{P}(x)) \vee (\exists x \mathcal{Q}(x)) \vdash \exists x(\mathcal{P}(x) \vee \mathcal{Q}(x))$         |

## Peano axioms

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• <math>\mathcal{A}_1 : \forall x \ S(x) \neq 0</math></li> <li>• <math>\mathcal{A}_2 : \forall x \ (x = 0) \vee (\exists y \ x = S(y))</math></li> <li>• <math>\mathcal{A}_3 : \forall x \ \forall y \ (S(x) = S(y)) \rightarrow (x = y)</math></li> <li>• <math>\mathcal{A}_4 : \forall x \ x + 0 = x</math></li> <li>• <math>\mathcal{A}_5 : \forall x \ \forall y \ x + S(y) = S(x + y)</math></li> <li>• <math>\mathcal{A}_6 : \forall x \ x \times 0 = 0</math></li> <li>• <math>\mathcal{A}_7 : \forall x \ \forall y \ x \times S(y) = (x \times y) + x</math></li> </ul> |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

And for every formula  $\mathcal{P}$  having  $x$  as a free variable:

$$\text{Rec}_{\mathcal{P}(x)} : \mathcal{P}(0) \wedge [\forall x \ \mathcal{P}(x) \rightarrow \mathcal{P}(S(x))] \rightarrow \forall x \ \mathcal{P}(x)$$

## Sources

The source files are available from:

[“GitHub: Logic”](#)

This book was translated by ChatGPT 5.6 from the original French version, which is available from:

[“GitHub: Logique”](#)

## Acknowledgments

I thank Stéphanie Bodin and Michel Bodin for proofreading the manuscript.



This book is distributed under the *Creative Commons – BY-NC-SA – 4.0* license.  
You can download the full-color book free of charge from the Exo7 website.

- T, 2
- F, 2
- 0, 2
- 1, 2
- $\neg$ , 2
- $\wedge$ , 2
- $\vee$ , 2
- $\rightarrow$ , 3
- $\leftrightarrow$ , 3
- $\equiv$ , 6
- $\top$ , 7
- $\perp$ , 7
- $\oplus$ , 9
- $\uparrow$ , 9
- $\models$ , 16
- $\vdash$ , 19
- $\forall$ , 51
- $\exists$ , 51
- $\models_{\mathcal{S}}$ , 74
- $\mathcal{P}[t/x]$ , 78
- $\mathcal{P}[x \mapsto d]$ , 78
- $\ulcorner \mathcal{P} \urcorner$ , 143
- $\bar{n}$ , 143
- alphabet, 5
- AND (logical), 2, 7
- arithmetic
  - elementary, 129
  - Peano, 130
  - Presburger, 134
  - Robinson, 134
- assignment, 74, 78
- assignment of truth values, 5, 42
- assumption, 16
- axioms, 127
  - Peano, 128
  - recursive, 179
- bounded quantification, 154
- bounded schema, 153
- Cantor's diagonal, 146
- Church's thesis, 179
- concatenation, 143
- conjunction, 2, 7
- connective, 4, 7, 9, 33, 52
- constant, 52
- continuum hypothesis, 123
- contraposition, 6, 26
- De Morgan's laws, 6, 26
- definable set, 133
- diagonalization, 145, 169
- disjunction, 2, 7
- domain, 64
- equality, 53
- equivalence, 3, 9
- evaluation, 5, 34, 42
- ex falso*, 22
- XOR exclusive or, 9, 36
- explosion, 22
- formula, 5, 41, 58
  - algebraic, 13
  - atomic, 54, 58
  - closed, 62, 74, 102
  - contradiction, 75
  - Gödel, 145, 177
  - height, 42
  - satisfaction, 10, 42, 74
  - validity, 75
- formula tree, 12
- fresh constant, 91
- function, 57
  - abstract, 33

- characteristic, 152
- Gödel's  $\beta$  function, 164
- primitive recursive, 149, 166
- recursive, 179
- representable, 160
- sign, 152

Gödel coding, 137, 156

Gödel number, 138, 139, 143

graph, 56

Henkin witness, 112

implication, 3, 8, 22

inference rules, 95

law of excluded middle, 7, 25

lemma

- diagonal, 170
- natural deduction, 103

logic

- classical, 23
- intuitionistic, 23
- minimal, 23

logic circuit, 11

logical consequence, 16, 43, 76

logical equivalence, 6, 76

model, 42, 74, 102, 109

*modus ponens*, 22

*modus tollens*, 26

NAND, 9, 36

negation, 2, 7

normal form

- conjunctive, 39
- disjunctive, 39

NOT (logical), 2, 7

numeral, 143

OR (logical), 2, 7

paradox

- barber, 144
- liar, 172
- Russell's, 144

Polish notation, 12

predicate, 52

premise, 16

proof, 19, 82

- Hilbert system, 157
- natural deduction, 20, 84

proof by contradiction, 26

quantifiers  $\forall, \exists$ , 51, 52, 73, 78, 82, 85

rules of inference, 21, 25, 29, 82

structure, 55, 64

substitution, 62, 78

tautology, 10

term, 54, 58

theorem, 19

- Chinese remainder theorem, 164
- closure, 105
- compactness, 43, 104
- completeness theorem, 30, 101, 109
- finiteness, 104
- incompleteness theorem, 121, 145, 174, 178
- second incompleteness theorem, 123, 180
- soundness, 94
- Tarski's, 172
- Tychonoff's, 47

theory, 102, 109

- complete, 110, 175
- consistent, 102, 109, 174, 180
- $\omega$ -consistent, 175
- set theory, 68

truncated subtraction, 150

truth table, 2, 5, 16

universal closure, 106

valuation, 42

variable, 52

- bound, 62
- free, 62, 74
- propositional, 2, 4
- scope, 61

Zermelo–Fraenkel, 135