Proofs in theories

Gilles Dowek

Proofs in theories

Gilles Dowek

September 2022

Contents

Introduction

Proofs and computer science

The logic of the thirties, culminating with Church's theorem on the undecidability of provability, has emphasized the difference between the notions of proof and algorithm. The method to judge a proposition true was then to build a proof and algorithms could only be used for very specific decidable problems. More recent works have emphasized the similarity and the links between theses two notions. These links are at least of four different kinds.

1. *Proof-checking and proof-search algorithms.* Although provability is undecidable, the correctness of a proof is decidable, hence it is possible to design proof-checking algorithms, and provability semi-decidable, hence it is possible to design proof-search semi-algorithms.

2. *Proofs of algorithms and programs.* Designing an algorithm and writing a program is such a difficult task that it is very difficult to avoid errors. A way to avoid such errors is to prove that the designed algorithm or program is correct.

3. *Constructivity and the Brouwer-Heyting-Kolmogorov interpretation.* Some proofs, called *constructive proofs*, can be defined as algorithms. Thus, the language of constructive proofs is a programming language. Moreover, in this language, all programs terminate.

4. *Theories.* Most proofs are not expressed in pure logic, but in a specific theory, such as arithmetic, set theory, type theory, etc. Theories can be defined as sets of axioms, but some of them can also be defined as algorithms.

These links between the notions of proof and algorithm have completely renewed proof-theory and made proof-theory one of the key chapters of theoretical computer science.

Proofs in theories

A proof is built with two kinds of ingredients: logical and theoretical ones. The deduction of the proposition B from the propositions A and $A \Rightarrow B$ is logical: this deduction

is possible whatever the propositions A and B speak about, because its validity only rests upon the meaning of the logical symbol $\Rightarrow$. In contrast, deducing the proposition $n = p$ from the proposition $n + 1 = p + 1$ is possible only in a specific theory: arithmetic, as it rests upon the meaning of the symbols $+$, 1, and $=$, that are specific to this theory. When a proof uses only logical ingredients, it is said to be *purely logical*. For instance, the proposition

$$2 + 2 = 4 \Rightarrow 2 + 2 = 4$$

has a purely logical proof, as its validity rests upon the meaning of the symbol $\Rightarrow$, and not on that of the symbols 2, 4, $+$, and $=$.

Proof-theory usually starts with the study of purely logical proofs, that are the simplest. Then, it usually focuses on proofs in some specific theories such as arithmetic—studied by Gentzen—or Simple type theory—studied by Girard. Sometimes this distinction between the logic and the theory vanishes in systems such as Martin-Löf's type theory or Coquand's and Huet's Calculus of constructions.

In these course notes, we have taken an approach to study proofs not in specific theories, such as arithmetic of Simple type theory, but an arbitrary theory. For instance, Theorem 9.1 and Corollaries 9.1, 9.2, 9.3, and 9.4 apply to any theory. The application to specific theories such as arithmetic and Simple type theory is only done in Corollaries 9.5 and 9.6. Thus, these course notes can be seen as a first sketch of a theory of theories. The first step towards such a theory is to change slightly the definition of the notion of theory: theories will not be defined, as usual, as set of axioms, but as algorithms.

Proof reduction and models

Key results in proof-theory are theorems establishing the termination of proof-reduction algorithms. These algorithms transform proofs into other proofs of the same proposition, by eliminating *cuts*. An example of *cut* is a sequence of deduction steps where, after having proved the propositions A and B, one deduces the proposition $A \wedge B$, and then the proposition A again. Such a cut can be eliminated, as one already has a proof of A.

A consequence of the termination proof-reduction algorithms is that any proposition that has a proof, also has a cut-free proof, that is a proof containing no cuts.

These termination of proof-reduction theorems are important because they have many corollaries. In proof-search, for instance, they permit to restrict the search to cut-free proofs, hence they are used to prove the completeness of proof-search methods, such as the Resolution method or the tableaux method. They are also used to prove the disjunction property and the witness property in various theories. Because constructive proofs are algorithms, the logic of constructive proofs in various theories are programming languages and the proof-reduction algorithms are the execution mechanisms of these programming language. The termination of proof-reduction theorems show that all programs terminate in these languages.

In these course notes, we have emphasized the link between proving the termination of proof reduction in a given theory and constructing a model of this theory. The notion

of model is a central notion in logic but, as we shall see, the models we need in proof-theory are slight generalization of the usual notion of model valued in $\{0, 1\}$, called *many-valued* models.

Structure of the document

These course notes are organized in four parts.

In Chapters 1, 2, and 3, we shall present the basic notions of proof, theory and model used in these course notes. When presenting the notion of proof we emphasize the notion of constructivity and that of cut. When we present the notion of theory, we emphasize that a theory should be defined as an algorithms rather than as a set of axioms. And when presenting the notion of model we emphasize the notion of many-valued models.

In Chapters 4, 5, 6, and 7, we present examples of theories, in particular arithmetic and Simple type theory.

In Chapters 8 and 9, we introduce proof reduction and model based methods to prove the termination of proof-reduction algorithms.

In Chapters 10, 11, and 12, we present formalisms where the notions of term, proposition and proof, kept separate in chapter 8 and 9, are unified into a single syntactic notion. This leads to introduce the $\lambda\Pi$-calculus, also known as λ-calculus with dependent types, and its extension the $\lambda\Pi$-calculus modulo theory. We then present two other formalisms: Martin-Löf's type theory and the Coquand's and Huet's Calculus of constructions, emphasizing the fact that although they have been defined before, these calculi can be seen as variants of the $\lambda\Pi$-calculus modulo theory, where Peano's predicate symbol is dropped or where propositions and propositional contents are identified.

The formalisms met during this journey are used in many proof-checking systems: Simple type theory is the base of the HOL, HOL-light, Isabelle/HOL, and PVS, the $\lambda\Pi$-calculus is the base of the Twelf system, the $\lambda\Pi$-calculus modulo theory is the base of the Dedukti system, Martin-Löf's type theory is the base of the Agda system, and the Calculus of constructions is the base of the Coq system.

Chapter 1

The Natural Deduction

1.1 Natural Deduction Rules

We assume the reader is familiar with the notion of inductive definition, the syntax of many-sorted Predicate logic—the notions of sort, language, term, proposition—, the notions of free and bound variables, alphabetic equivalence and substitution, and the basis of computability theory—the expression of computable functions as in arithmetic, in the language of rewrite rules and in the λ-calculus.

We shall use a formulation of Predicate logic with the connectives $\top$, $\bot$, $\wedge$, $\vee$, and $\Rightarrow$ and the quantifiers $\forall$ and $\exists$. The connective $\neg$ is not primitive and the proposition $\neg A$ is an abbreviation for $A \Rightarrow \bot$. In the same way, the proposition $A \Leftrightarrow B$ is an abbreviation for $(A \Rightarrow B) \wedge (B \Rightarrow A)$.

Our goal in this section is to define the notion of *provable proposition*. It is usual to define this notion inductively with deduction rules such as the rule

$$\frac{A \Rightarrow B \quad A}{B}$$

that expresses that if the propositions $A \Rightarrow B$ and A have already been proved, then the proposition B can be proved as well. Yet, such rules force us to use the same hypotheses for the whole proof. It is hard to translate a standard reasoning pattern: to prove $A \Rightarrow B$, assume A and prove B under this hypothesis. This observation led to the introduction of a notion of pair, formed with a finite set of hypotheses and a conclusion. Such a pair is called a *sequent*.

Definition 1.1 (Sequent) *A* sequent *is a pair $\Gamma \vdash A$, where Γ is a finite set of propositions and A is a proposition.*

The rules of Natural deduction are deduction rules allowing to deduce sequents from sequents.

Definition 1.2 (Natural deduction rules)

$$\frac{}{\Gamma \vdash A}\ \textit{axiom if } A \in \Gamma$$

$$\frac{}{\Gamma \vdash \top}\ \top\textit{-intro}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash A}\ \bot\textit{-elim}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}\ \wedge\textit{-elim}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}\ \wedge\textit{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}\ \vee\textit{-intro}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}\ \vee\textit{-intro}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\textit{-elim}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\textit{-intro}$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\textit{-elim}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall x\ A}\ \forall\textit{-intro if } x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \forall x\ A}{\Gamma \vdash (t/x)A}\ \forall\textit{-elim}$$

$$\frac{\Gamma \vdash (t/x)A}{\Gamma \vdash \exists x\ A}\ \exists\textit{-intro}$$

$$\frac{\Gamma \vdash \exists x\ A \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ \exists\textit{-elim if } x \notin FV(\Gamma, B)$$

$$\frac{}{\Gamma \vdash A \vee \neg A}\ \textit{excluded-middle}$$

Remark. Except for the *axiom* rule and the *excluded-middle* rule, each rule is about a single connective or quantifier. This provides a first classification of rules: there are Natural deduction rules for the conjunction $\wedge$, for the disjunction $\vee$, etc. Then, among the rules of a specific connective or quantifier, we can distinguish the rules where this connective or quantifiers is in the conclusion of the rule and those where it is in a premise. The former are called *introduction* rules and the latter *elimination* rules. For instance the conjunction $\wedge$ has one introduction rule

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge\text{-intro}$$

and two elimination rules

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge\text{-elim}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge\text{-elim}$$

The introduction rule expresses what has to be done to prove a proposition $A \wedge B$: proving A and then proving B. The elimination rules express what can be done once the proposition $A \wedge B$ has been proved: deduce A and deduce B.

Definition 1.3 (Provable sequent, proof of a sequent) *The set of* provable sequents *is inductively defined by the Natural deduction rules.*

A proof *of a sequent $\Gamma \vdash A$ is a derivation of this sequent using the rules of Natural deduction, that is, a tree where nodes are labelled by sequents and where the root is labelled by $\Gamma \vdash A$, and such that if a node is labelled by a sequent $\Delta \vdash B$ and its children are labelled by sequents $\Sigma_1 \vdash C_1, \ldots, \Sigma_n \vdash C_n$ then there is a Natural deduction rule that deduces $\Delta \vdash B$ from $\Sigma_1 \vdash C_1, \ldots, \Sigma_n \vdash C_n$.*

Definition 1.4 (Provable proposition, proof of a proposition) *A proposition A is said to be* provable in Predicate logic*, or* provable without any axioms*, if the sequent $\vdash A$—with no propositions on the left of the sign $\vdash$—is provable. A* proof *of this proposition is a proof of the sequent $\vdash A$.*

Exercise 1.1 *Consider a language with three sorts of terms:* point*,* line *and* scalar*, two predicate symbols $=$ with arity $\langle \text{scalar}, \text{scalar} \rangle$ and $\in$ with arity $\langle \text{point}, \text{line} \rangle$ and two function symbols d,* distance*, with arity $\langle \text{point}, \text{point}, \text{scalar} \rangle$ and b,* bisector*, with arity $\langle \text{point}, \text{point}, \text{line} \rangle$. Let Γ be the set containing the propositions*

$$\forall x \forall y \forall z \; (x \in b(y, z) \Leftrightarrow d(x, y) = d(x, z))$$

$$\forall x \forall y \forall z \; ((x = y \wedge y = z) \Rightarrow x = z)$$

and A a proposition stating that if two bisectors of the triangle xyz intersect at a point w, then the three bisectors intersect at this point:

$$\forall w \forall x \forall y \forall z \; ((w \in b(x, y) \wedge w \in b(y, z)) \Rightarrow w \in b(x, z))$$

Write a proof of the sequent $\Gamma \vdash A$.

Proposition 1.1 (Weakening) *If the sequent $\Gamma \vdash A$ is provable, then the sequent $\Gamma, B \vdash A$ also is provable.*

Proof. By induction over the structure of a proof of $\Gamma \vdash A$.

Definition 1.5 (Axiomatic theory) *A* axiomatic theory *$\mathcal{T}$ is a finite or infinite set of closed propositions called* axioms*.*

Definition 1.6 (Proof in a theory) *Let $\mathcal{T}$ be an axiomatic theory and A be a proposition. The proposition A is provable in the theory $\mathcal{T}$ if there exists a finite subset Γ of $\mathcal{T}$ such that the sequent $\Gamma \vdash A$ is provable.*

1.2 Constructive proofs

Let P be a set that contains 0 but not 2. We can prove that there exists a natural number x such that x is in P and $x+1$ is not. The usual argument, that the sequence of natural numbers must quit the set P at some point between 0 and 2 boils down to the following:

- either 1 is in P or 1 is not in P,
- in the first case, 1 is in P and 2 is not,
- in the second, 0 is in P and 1 is not.

In Natural deduction, this leads to the following proof of the sequent

$$\Gamma \vdash \exists x\ (P(x) \wedge \neg P(S(x)))$$

where $\Gamma = \{P(0), \neg P(S(S(0)))\}$

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash P(S(0)) \vee \neg P(S(0))} \quad \dfrac{\dfrac{\dfrac{}{\Gamma,\, P(S(0)) \vdash P(S(0))} \quad \dfrac{}{\Gamma,\, P(S(0)) \vdash \neg P(S(S(0)))}}{\Gamma,\, P(S(0)) \vdash P(S(0)) \wedge \neg P(S(S(0)))}}{\Gamma,\, P(S(0)) \vdash \exists x\ (P(x) \wedge \neg P(S(x)))} \quad \dfrac{\dfrac{\dfrac{}{\Gamma,\, \neg P(S(0)) \vdash P(0)} \quad \dfrac{}{\Gamma,\, \neg P(S(0)) \vdash \neg P(S(0))}}{\Gamma,\, \neg P(S(0)) \vdash P(0) \wedge \neg P(S(0))}}{\Gamma,\, \neg P(S(0)) \vdash \exists x\ (P(x) \wedge \neg P(S(x)))}}{\Gamma \vdash \exists x\ (P(x) \wedge \neg P(S(x)))}$$

Yet, this proof of the proposition

$$\exists x\ (P(x) \wedge \neg P(S(x)))$$

does not provide any number n such that the proposition $P(n) \wedge \neg P(S(n))$ is provable from the axioms $P(0)$ and $\neg P(S(S(0)))$. Moreover, it is not difficult to prove that, for every number n, the proposition $P(n) \wedge \neg P(S(n))$ is not provable from the axioms $P(0)$ and $\neg P(S(S(0)))$.

To avoid the axioms $P(0)$ and $\neg P(S(S(0)))$, let us transform the proposition into

$$\exists x\ (P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(x) \wedge P(S(x))))$$

A similar argument shows that this proposition A has a proof without any axioms. As above, we first build a proof π_1 of the sequent $P(S(0)) \vdash A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{P(S(0)), P(0), \neg P(S(S(0))) \vdash P(S(0))} \quad \dfrac{}{P(S(0)), P(0), \neg P(S(S(0))) \vdash \neg P(S(S(0)))}}{P(S(0)), P(0), \neg P(S(S(0))) \vdash P(S(0)) \wedge \neg P(S(S(0)))}}{P(S(0)), P(0) \vdash \neg P(S(S(0))) \Rightarrow (P(S(0)) \wedge \neg P(S(S(0))))}}{P(S(0)) \vdash P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(S(0)) \wedge \neg P(S(S(0))))}}{P(S(0)) \vdash \exists x\ (P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(x) \wedge \neg P(S(x))))}$$

and a proof π_2 of the sequent $\neg P(S(0)) \vdash A$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg P(S(0)), P(0), \neg P(S(S(0))) \vdash \neg P(S(0))} \quad \dfrac{}{\neg P(S(0)), P(0), \neg P(S(S(0))) \vdash P(0)}}{\neg P(S(0)), P(0), \neg P(S(S(0))) \vdash P(0) \wedge \neg P(S(0))}}{\neg P(S(0)), P(0) \vdash \neg P(S(S(0))) \Rightarrow (P(0) \wedge \neg P(S(0)))}}{\neg P(S(0)) \vdash P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(0) \wedge \neg P(S(0)))}}{\neg P(S(0)) \vdash \exists x\ (P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(x) \wedge \neg P(S(x))))}$$

And we glue these two sub-proofs together with the $\vee$-elim rule

$$\dfrac{\dfrac{}{\vdash P(S(0)) \vee \neg P(S(0))} \quad \dfrac{\overset{\pi_1}{}}{P(S(0)) \vdash A} \quad \dfrac{\overset{\pi_2}{}}{\neg P(S(0)) \vdash A}}{\vdash A}$$

But again, for every number n, the proposition

$$P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(n) \wedge P(S(n)))$$

is not provable.

A set E of propositions is said *to have the witness property* if whenever a proposition of the form $\exists x\ A$ is in E, then there exists a term t such that the proposition $(t/x)A$ is in E as well. And the term t is said to be a *witness* of the proposition $\exists x\ A$ in E. The example above shows that the set of provable propositions in Predicate logic does not have the witness property.

Analyzing the proof, we can understand how the witness is lost. In the sub-proof π_1 a witness is given: $S(0)$. And in the sub-proof π_2 also a witness is given: 0. The witness is lost when these two sub-proofs are glued together with the $\vee$-elim rule. The proposition A has been proved under the hypothesis $P(S(0))$ and under the hypothesis $\neg P(S(0))$, thus, we can conclude that it holds in both cases. But, as we do not not know whether $P(S(0))$ or $\neg P(S(0))$ holds, we have no way to chose between the two witnesses $S(0)$ and 0 given in the two sub-proofs.

In order to use the $\vee$-elim rule, we need to prove a third proposition: $P(S(0)) \vee \neg P(S(0))$. This proposition is proved in the leftmost branch of the proof with the *excluded-middle* rule. This rules states that the proposition $A \vee \neg A$ always holds, even if we do not know which of the propositions A and $\neg A$ does. This rule seems to be at the origin of the loss of the witness property. A proof that does not use this rule is called *a constructive proof*.

Definition 1.7 (Constructive proof, classical proof) *A proof is* constructive *if it does not use the* excluded-middle *rule. In contrast, a proof that may use the* excluded-middle *rule is called* classical.

As we shall see, the set of propositions that have a constructive proof without any axioms has the witness property. This witness property also extends to some theories, but not all. This shows, among other things, that the proposition

$$\exists x\ (P(0) \Rightarrow \neg P(S(S(0))) \Rightarrow (P(x) \wedge \neg P(S(x))))$$

does not have a constructive proof without any axioms. Moreover there is an algorithm—as we shall see: a proof-reduction algorithm—to extract a witness from an existence proof.

This witness property makes the language of constructive proofs a programming language, and the witness extraction algorithm the execution mechanism of this programming language. For instance, assume the proposition

$$\forall x \exists y\ (x = 2 \times y \vee x = 2 \times y + 1)$$

has a constructive proof π in some theory, that the set of propositions that have a constructive proof in this theory has the witness property, and that we have an algorithm to extract a witness from an existence a proof in this theory. From the proof π, it is easy to build a proof of the proposition

$$\exists y\ (25 = 2 \times y \vee 25 = 2 \times y + 1)$$

with the $\forall$-elim rule

$$\dfrac{\dfrac{\pi}{\Gamma \vdash \forall x \exists y\ (x = 2 \times y \vee x = 2 \times y + 1)}}{\exists y\ (25 = 2 \times y \vee 25 = 2 \times y + 1)}$$

from this proof we can extract a witness t. It is not difficult to see that, as the proposition $25 = 2 \times t \vee 25 = 2 \times t + 1$ is provable, this witness can only be 12. Thus, applying the proof π to a number n and extracting the witness from the obtained existence proof is a way to divide the number n by 2. Therefore, the proof π is a program that divides its argument by 2. Moreover, by construction, this program is correct with respect to the specification

$$x = 2 \times y \vee x = 2 \times y + 1$$

1.3 Cuts and proof reduction

1.3.1 Cuts

Definition 1.8 (Cut) *A cut is a proof ending with an elimination rule whose main premise is proved by an introduction rule on the same symbol. The different cases are*

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A} \quad \dfrac{\pi'}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}}{\Gamma \vdash A}\ \wedge\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A} \quad \dfrac{\pi'}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}}{\Gamma \vdash B}\ \wedge\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro} \quad \dfrac{\pi'}{\Gamma, A \vdash C} \quad \dfrac{\pi''}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ \vee\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash B}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro} \quad \dfrac{\pi'}{\Gamma, A \vdash C} \quad \dfrac{\pi''}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ \vee\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\Rightarrow\textit{-intro} \quad \dfrac{\pi'}{\Gamma \vdash A}}{\Gamma \vdash B}\Rightarrow\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash \forall x\ A}\forall\textit{-intro}}{\Gamma \vdash (t/x)A}\forall\textit{-elim}$$

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash (t/x)A}}{\Gamma \vdash \exists x\ A}\exists\textit{-intro} \quad \dfrac{\pi'}{\Gamma, A \vdash B}}{\Gamma \vdash B}\exists\textit{-elim}$$

A proof contains a cut *if one of its sub-trees is a cut. Otherwise, it is* cut-free.

1.3.2 Properties of cut-free proofs

Proposition 1.2 (Final rule) *If π is a proof of a proposition A that is (1.) constructive, (2.) cut-free, and (3.) without any axioms, then it ends with an introduction rule.*

Proof. As the proof π is a proof of the proposition A without any axioms, it is a proof of the sequent $\vdash A$. By induction over the structure of this proof, we show that it ends with an introduction rule. As the proof is constructive, it ends either with an axiom rule, an elimination rule or an introduction rule. The last rule cannot be an axiom rule, because the left-hand part of the sequent is empty. If this last rule is an elimination, then the proof of the main premise of this elimination rule is constructive, cut-free and it proves a sequent whose left-hand part is empty. Hence, by induction hypothesis, it ends with an introduction and the proof is a cut, contradicting the fact that it is cut-free.

Proposition 1.3 (Disjunction property) *If a proposition $A \vee B$ has a proof that is (1.) constructive, (2.) cut-free, and (3.) without any axioms, then there exists a proof of A or there exists a proof of B, and this proof is constructive, cut-free, and without any axioms.*

Proof. The proof ends with a $\vee$-intro rule, whose immediate sub-tree is a proof of A or a proof of B.

Proposition 1.4 (Witness property) *If a proposition $\exists x\ A$ has a proof that is (1.) constructive, (2.) cut-free, and (3.) without any axioms, then there exists a term t and a proof of $(t/x)A$ that is constructive, cut-free, and without any axioms.*

Proof. The proof ends with a $\exists$-intro rule, whose immediate sub-tree is a proof of $(t/x)A$ for some term t.

1.3.3 Proof reduction

This witness property motivates the project to define an algorithm transforming each proof into a cut-free proof. As we shall see, when a proof contains a cut, it is not difficult to eliminate it. But, as eliminating a cut may create other cuts, the difficulty is in proving the termination of this process.

Definition 1.9 (Proof reduction) *We define a process that eliminates cuts step by step in proofs. A cut of the form*

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A} \quad \dfrac{\pi'}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}}{\Gamma \vdash A}\ \wedge\textit{-elim}$$

is replaced by the proof π. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A} \quad \dfrac{\pi'}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}}{\Gamma \vdash B}\ \wedge\textit{-elim}$$

is replaced by the proof π'. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro} \quad \dfrac{\pi'}{\Gamma, A \vdash C} \quad \dfrac{\pi''}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ \vee\textit{-elim}$$

is replaced by the proof obtained this way: in the proof π' we remove the hypothesis A in all sequents, then each time the axiom rule is used with this proposition, we replace it by the proof π. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash B}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro} \quad \dfrac{\pi'}{\Gamma, A \vdash C} \quad \dfrac{\pi''}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ \vee\textit{-elim}$$

is replaced by the proof obtained this way: in the proof π'' we suppress the hypothesis B in all sequents, then each time the axiom rule is used with this proposition, we replace it by the proof π. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\textit{-intro} \quad \dfrac{\pi'}{\Gamma \vdash A}}{\Gamma \vdash B}\ \Rightarrow\textit{-elim}$$

is replaced by the proof obtained this way: in the proof π we suppress the hypothesis A in all sequents, then each time the axiom rule is used with this proposition, we replace

it with the proof π'. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash \forall x\ A}\,\forall\textit{-intro}}{\Gamma \vdash (t/x)A}\,\forall\textit{-elim}$$

is replaced by the proof π where the variable x is substituted by the term t everywhere. A cut of the form

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash (t/x)A}}{\Gamma \vdash \exists x\ A}\,\exists\textit{-intro}\quad \dfrac{\pi'}{\Gamma, A \vdash B}}{\Gamma \vdash B}\,\exists\textit{-elim}$$

is replaced by the proof obtained this way: in the proof π', we substitute the variable x by the term t everywhere, then we suppress the hypothesis $(t/x)A$ in all sequents and each time the axiom rule is used with this proposition, we replace it with the proof π.

Thus, when a proof contains a cut, it is not difficult to eliminate it. But, as already said, eliminating a cut may create new cuts. For instance, let $\Gamma = P, P \Rightarrow Q$, the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, Q \vdash Q}\,\text{axiom}\quad \dfrac{}{\Gamma, Q \vdash Q}\,\text{axiom}}{\Gamma, Q \vdash Q \wedge Q}\,\wedge\text{-intro}}{\Gamma \vdash Q \Rightarrow (Q \wedge Q)}\,\Rightarrow\text{-intro}\quad \dfrac{}{\Gamma \vdash \top}\,\top\text{-intro}}{\Gamma \vdash (Q \Rightarrow (Q \wedge Q)) \wedge \top}\,\wedge\textbf{-intro}}{\Gamma \vdash Q \Rightarrow (Q \wedge Q)}\,\wedge\textbf{-elim}\quad \dfrac{\dfrac{}{\Gamma \vdash P \Rightarrow Q}\,\text{axiom}\quad \dfrac{}{\Gamma \vdash P}\,\text{axiom}}{\Gamma \vdash Q}\,\Rightarrow\text{-elim}}{\Gamma \vdash Q \wedge Q}\,\Rightarrow\text{-elim}$$

contains a cut, formed with the rules $\wedge$-intro and $\wedge$-elim, and eliminating this cut yields the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, Q \vdash Q}\,\text{axiom}\quad \dfrac{}{\Gamma, Q \vdash Q}\,\text{axiom}}{\Gamma, Q \vdash Q \wedge Q}\,\wedge\text{-intro}}{\Gamma \vdash Q \Rightarrow (Q \wedge Q)}\,\Rightarrow\textbf{-intro}\quad \dfrac{\dfrac{}{\Gamma \vdash P \Rightarrow Q}\,\text{axiom}\quad \dfrac{}{\Gamma \vdash P}\,\text{axiom}}{\Gamma \vdash Q}\,\Rightarrow\text{-elim}}{\Gamma \vdash Q \wedge Q}\,\Rightarrow\textbf{-elim}$$

that contains a cut, formed with the rules $\Rightarrow$-intro and $\Rightarrow$-elim, that was not present in the original one. So the key question we shall discuss in Chapter 8 is that of the termination of this proof-reduction process. When we have proved termination of proof reduction, we shall conclude that a sequent has a proof if and only if it has a cut-free proof and we shall be able to extend the witness property to all constructive proofs without any axioms, whether they are cut-free or not. The extraction of a witness from a will be based on proof reduction.

Exercise 1.2 *Eliminate the cuts in the proof*

$$\dfrac{\dfrac{\dfrac{}{\exists x\,(P(x) \Rightarrow P(x)) \vdash \exists x\,(P(x) \Rightarrow P(x))}\,\text{axiom}}{\vdash \exists x\,(P(x) \Rightarrow P(x)) \Rightarrow \exists x\,(P(x) \Rightarrow P(x))}\,\Rightarrow\text{-intro}\quad \dfrac{\dfrac{\dfrac{}{P(c) \vdash P(c)}\,\text{axiom}}{\vdash P(c) \Rightarrow P(c)}\,\Rightarrow\text{-intro}}{\vdash \exists x\,(P(x) \Rightarrow P(x))}\,\exists\text{-intro}}{\vdash \exists x\,(P(x) \Rightarrow P(x))}\,\Rightarrow\text{-elim}$$

Exercise 1.3 *Find a proof π that contains a single cut but such that eliminating this cut in π creates other cuts.*

Chapter 2

The Deduction modulo a congruence

In Chapter 1, we have seen that if a proof π is (1) constructive, (2) cut-free, and (3) without any axioms, then it ends with an introduction rule (Proposition 1.2). As corollaries, we got the disjunction property (Proposition 1.3): if there exists a proof of $A \vee B$ then there exists either a proof of A or a proof of B and the witness property (Proposition 1.4): if there exists a proof of $\exists x\ A$ then there exists a term t and a proof of the proposition $(t/x)A$—this witness property being the key property to use proofs as programs. The condition (2) will not be a restriction when we have proved the termination of proof-reduction. The condition (1) is a restriction, but not a severe one: many theorems have a constructive proof. But the condition (3) is a severe restriction: without any axioms we cannot prove much. One of the goals of proof-theory is to extend the final rule property and the witness property to proofs in theories.

The property that constructive cut-free proofs end with an introduction rule does not generalize to all theories. For instance, the proof

$$\frac{}{\exists x\ P(x) \vdash \exists x\ P(x)}\ \text{axiom}$$

is a proof of the proposition $\exists x\ P(x)$ in the theory formed with the the axiom $\exists x\ P(x)$ and it ends, not with an introduction rule, but with an axiom rule. This example also shows that the witness property does not extend to all theories.

Another example: imagine that we want to use an abbreviation 1 for the the term $S(0)$. How can we use this definition $1 = S(0)$ in a proof? A first solution is to introduce a constant 1 and an axiom $1 = S(0)$. This axiom jeopardizes the final rule property. For instance, the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash \forall x \forall y\ (x = y \Rightarrow P(x) \Rightarrow P(y))}\ \text{axiom}}{\Gamma \vdash \forall y\ (1 = y \Rightarrow P(1) \Rightarrow P(y))}\ \forall\text{-elim}}{\Gamma \vdash 1 = S(0) \Rightarrow P(1) \Rightarrow P(S(0))}\ \forall\text{-elim} \qquad \dfrac{}{\Gamma \vdash 1 = S(0)}\ \text{axiom}}{\Gamma \vdash P(1) \Rightarrow P(S(0))}\ \Rightarrow\text{-elim}$$

where $\Gamma = \{1 = S(0), \forall x \forall y\ (x = y \Rightarrow P(x) \Rightarrow P(y))\}$ is a proof of the proposition $P(1) \Rightarrow P(S(0))$ in the theory formed with the axiom $1 = S(0)$ and also the axiom $\forall x \forall y\ (x = y \Rightarrow P(x) \Rightarrow P(y))$. It is cut free, but ends with an elimination rule.

Another approach to the notion of definition is to consider that each time we read the symbol 1, we mentally replace it by the term $S(0)$. This second approach does not jeopardize the witness property, as the proof above can be reformulated as

$$\dfrac{\dfrac{}{P(1) \vdash P(S(0))}\ \text{axiom}}{\vdash P(1) \Rightarrow P(S(0))}\ \Rightarrow\text{-into}$$

that uses no axioms and hence ends with an introduction rule. But in the axiom rule, we must accept to mentally replace 1 by $S(0)$. This amounts to say that deduction is performed modulo an equivalence relation identifying the terms 1 and $S(0)$.

Besides definitions, we are used to consider such equivalence relations, when we reason informally, for instance with an associative operation. Instead of using an axiom

$$\forall x \forall y \forall z\ ((x + y) + z = x + (y + z))$$

we just consider $(t + u) + v$ and $t + (u + v)$ as equivalent terms, and sometimes even write this term $t + u + v$ without any parentheses.

On the other hand, we cannot identify too many propositions. For instance, if we identified all the provable propositions with $\top$, then all provable propositions, including existential ones, would have a trivial proof

$$\dfrac{}{\vdash A}\ \top\text{-intro}$$

In particular, for the correctness of a proof to be decidable, we must restrict to a decidable equivalence relation. As it is decidable, this equivalence relation handles the computation part of proofs, while the deduction rules handle the deduction part.

The congruence also needs to be *non confusing*, for instance, it should not identify a proposition of the form $\exists x\ A$ and $\top$, because then a proof of $\exists x\ A$, even if it ends with an introduction rule, could fail to end with an $\exists$-intro rule: it could, for instance, end with a $\top$-intro rule. Then, the final rule property may fail to imply the witness property. The congruence should also not identify a proposition of the form $A \vee B$ with a proposition of the form $C \wedge D$. Otherwise, we could have, in a proof, a sequence of a $\vee$-introduction rule followed by a $\wedge$-elim rule

$$\dfrac{\dfrac{\dfrac{\dots}{\vdash A}}{\vdash C \wedge D}\ \vee\text{-intro}}{\vdash C}\ \wedge\text{-elim}$$

and there is no obvious way to reduce such a cut.

2.1 Deduction rules

Definition 2.1 (Congruence) *An equivalence relation relating terms to terms and propositions to propositions is a* congruence *if*

- *if* $t_1 \equiv t'_1$, ..., $t_n \equiv t'_n$ *then* $f(t_1, ..., t_n) \equiv f(t'_1, ..., t'_n)$,
- *if* $t_1 \equiv t'_1$, ..., $t_n \equiv t'_n$ *then* $P(t_1, ..., t_n) \equiv P(t'_1, ..., t'_n)$,
- *if* $A \equiv A'$ *and* $B \equiv B'$ *then* $(A \wedge B) \equiv (A' \wedge B')$, $(A \vee B) \equiv (A' \vee B')$, $(A \Rightarrow B) \equiv (A' \Rightarrow B')$,
- *if* $A \equiv A'$ *then* $(\forall x\ A) \equiv (\forall x\ A')$, *and* $(\exists x\ A) \equiv (\exists x\ A')$.

Definition 2.2 (Non confusing) *A congruence is said to be* non confusing *if, whenever* A *and* A' *are two propositions such that* $A \equiv A'$, *either* A *or* A' *is atomic or*

- $A = A' = \top$,
- $A = A' = \bot$,
- $A = B \wedge C$, $A' = B' \wedge C'$, $B \equiv B'$, *and* $C \equiv C'$,
- $A = B \vee C$, $A' = B' \vee C'$, $B \equiv B'$, *and* $C \equiv C'$,
- $A = B \Rightarrow C$, $A' = B' \Rightarrow C'$, $B \equiv B'$, *and* $C \equiv C'$,
- $A = \forall x\ B$, $A' = \forall x\ B'$, *and* $B \equiv B'$,
- *or* $A = \exists x\ B$ *and* $A' = \exists x\ B'$, *and* $B \equiv B'$.

Definition 2.3 (Rules of Natural Deduction modulo theory) *Let us consider a decidable and non confusing congruence* $\equiv$. *Then the rules of Natural deduction modulo* $\equiv$ *are the same as those of Natural deduction except that each proposition can be replaced at anytime with an equivalent proposition.*

$$\frac{}{\Gamma \vdash B}\ \textit{axiom if } A \in \Gamma \textit{ and } A \equiv B$$

$$\frac{}{\Gamma \vdash A}\ \top\textit{-intro if } A \equiv \top$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A}\ \bot\textit{-elim if } B \equiv \bot$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash C}\ \wedge\textit{-intro if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash A}\ \wedge\textit{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B}\ \wedge\textit{-elim if } C \equiv (A \wedge B)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash C}\ \vee\textit{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C}\ \vee\textit{-intro if } C \equiv (A \vee B)$$

$$\frac{\Gamma \vdash D \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}\ \vee\textit{-elim if } D \equiv (A \vee B)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash C}\ \Rightarrow\textit{-intro if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash C \quad \Gamma \vdash A}{\Gamma \vdash B}\ \Rightarrow\textit{-elim if } C \equiv (A \Rightarrow B)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash B}\ \langle x, A\rangle\ \forall\textit{-intro if } B \equiv (\forall x\ A) \textit{ and } x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash C}\ \langle x, A, t\rangle\ \forall\textit{-elim if } B \equiv (\forall x\ A) \textit{ and } C \equiv (t/x)A$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash B}\ \langle x, A, t\rangle\ \exists\textit{-intro if } B \equiv (\exists x\ A) \textit{ and } C \equiv (t/x)A$$

$$\frac{\Gamma \vdash C \quad \Gamma, A \vdash B}{\Gamma \vdash B}\ \langle x, A\rangle\ \exists\textit{-elim if } C \equiv (\exists x\ A) \textit{ and } x \notin FV(\Gamma, B)$$

Definition 2.4 (Provable sequent, proof of a sequent) *The set of sequents provable modulo a congruence $\equiv$ is inductively defined by the rules of Deduction modulo theory. A* proof *of a sequent $\Gamma \vdash A$ modulo a congruence $\equiv$ is a derivation of this sequent using the rules of Deduction modulo theory and the congruence $\equiv$.*

Definition 2.5 (Theory in Deduction modulo theory) *In Deduction modulo theory, a* theory *is formed with a set $\mathcal{T}$ of closed propositions called* axioms *and a decidable and non confusing congruence $\equiv$.*

Definition 2.6 (Provability in a theory) *The proposition A is provable in a theory $\mathcal{T}, \equiv$, if there exists a finite subset Γ of $\mathcal{T}$ such that the sequent $\Gamma \vdash A$ has a proof modulo $\equiv$.*

Definition 2.7 (Purely computational theory) *A theory $\mathcal{T}, \equiv$ is said to be* purely computational *if $\mathcal{T} = \varnothing$.*

Definition 2.8 (Purely axiomatic theory) *A theory $\mathcal{T}, \equiv$ is said to be* purely axiomatic *if the congruence $\equiv$ is the identity.*

Example 2.1 *Consider a decidable and non confusing congruence $\equiv$ such that $(2 \times 2 = 4) \equiv \top$. Then, in the theory $\varnothing, \equiv$, the number 4 can be proved to be even with the proof*

$$\frac{\dfrac{}{\vdash 2 \times 2 = 4}\ \top\textit{-intro}}{\vdash \exists x\ (2 \times x = 4)}\ \langle x, 2 \times x = 4, 2\rangle\ \exists\textit{-intro}$$

Proposition 2.1 (Equivalence) *For every theory $\mathcal{T}, \equiv$ there is a purely axiomatic theory, that is a set of axioms $\mathcal{T}'$, such that a proposition A is provable in $\mathcal{T}, \equiv$ if and only if it is provable in $\mathcal{T}'$.*

Proof. Take, for instance, all the axioms of $\mathcal{T}$ and those of the form $\forall x_1 \ldots \forall x_n \ (A \Leftrightarrow B)$ where $A \equiv B$.

2.2 Congruences defined with reduction rules

Congruences used in Deduction modulo theory are often defined with *reduction rules*, also called *rewrite rules*. For instance a congruence such that $(2 \times 2 = 4) \equiv \top$ may be defined with the following reduction rules

$$0 + y \longrightarrow y$$
$$S(x) + y \longrightarrow S(x + y)$$
$$0 \times y \longrightarrow 0$$
$$S(x) \times y \longrightarrow x \times y + y$$
$$0 = 0 \longrightarrow \top$$
$$S(x) = 0 \longrightarrow \bot$$
$$0 = S(y) \longrightarrow \bot$$
$$S(x) = S(y) \longrightarrow x = y$$

2.2.1 Reduction rules and congruences

Definition 2.9 (Reduction rule, reduction system) *A* reduction rule *is an ordered pair of terms or an ordered pair of propositions $\langle l, r \rangle$ written $l \longrightarrow r$. A* reduction system *is a set of reduction rules.*

Definition 2.10 (One step reduction at the root) *Let $\mathcal{R}$ be a reduction system. A term (resp. a proposition) t reduces in one step at the root to a term (resp. a proposition) u is there exists a substitution σ such that $t = \sigma l$ and $u = \sigma r$.*

Definition 2.11 (One step reduction) *Let $\mathcal{R}$ be a reduction system. A term (resp. a proposition) t reduces to a term (resp. a proposition) u in one step ($t \longrightarrow^1 u$) if there exists a sub-expression t' at occurrence q in t and a substitution σ such that $t' = \sigma l$ and the term u (resp. the proposition) is obtained by replacing in t the sub-expression t' at q by σr.*

Definition 2.12 (Reducible at the root, reducible, irreducible) *Let $\mathcal{R}$ be a reduction system and t be a term (resp. a proposition). The term (resp. the proposition) t is* reducible at the root *if there exists a term u such that t reduces to u in one step at the root, that is if there exists a rule $l \longrightarrow r$ in $\mathcal{R}$ and a substitution σ such that $t = \sigma l$.*

A term (resp. a proposition) t is reducible *if there exists a term u such that t reduces to u in one step, that is if a sub-expression of t is reducible at the root. It is* irreducible *otherwise.*

From the the relation $\longrightarrow^1$ we can define more relations: $\longrightarrow^*$, $\longrightarrow^+$, and $\equiv$. Note that all these definitions, as well as the notions of termination and congruence below, can be applied to any binary relation $\longrightarrow^*$, whatever its definition is.

Definition 2.13 (Reduction sequence) *Let $\mathcal{R}$ be a reduction system. A* reduction sequence *is a finite or infinite sequence of terms (resp. of propositions) $t_0, t_1, ...$ such that for every i, $t_i \longrightarrow^1 t_{i+1}$.*

Definition 2.14 (Reduction, reduction in at least one step) *Let $\mathcal{R}$ be a reduction system. A term (resp. a proposition) t* reduces *to a term (resp. a proposition) u ($t \longrightarrow^* u$) if there exists a finite reduction sequence starting on t and ending on u. A term (resp. a proposition) t* reduces in at least one step *to a term (resp. a proposition) u ($t \longrightarrow^+ u$) if there exists a term (resp. a proposition) v such that $t \longrightarrow^1 v \longrightarrow^* u$.*

Definition 2.15 (Irreducible form) *A term u is a* irreducible form *of a term t if $t \longrightarrow^* u$ and u is irreducible.*

Definition 2.16 (Congruence sequence) *Let $\mathcal{R}$ be a reduction system. A* congruence sequence *is a finite or infinite sequence of terms (resp. of propositions) $t_0, t_1, ...$ such that for every i, $t_i \longrightarrow^1 t_{i+1}$ or $t_{i+1} \longrightarrow^1 t_i$.*

Definition 2.17 (Congruence) *Let $\mathcal{R}$ be a reduction system. Two terms (resp. two propositions) t and u are* congruent *($t \equiv u$) if there exists a finite congruence sequence starting on t and ending on u.*

2.2.2 Decidability

The relation $\equiv$ defined by any reduction system is an equivalence relation by construction. It is also a congruence by construction, as Definition 2.11 permits to reduce any sub-expression. More properties are needed for this relation to be decidable and non confusing.

Definition 2.18 (Termination, strong termination) *A term (resp. a proposition) t* terminates *if it has a irreducible form, that is if there exists a finite reduction sequence starting on t and ending on a irreducible expression. The term (resp. the proposition) t* strongly terminates *if all reduction sequences starting from t are finite. A reduction system terminates (resp. strongly terminates) if all terms and all propositions terminate (resp. strongly terminate).*

Definition 2.19 (Confluent) *A reduction system is* confluent *if whenever a term (resp. proposition) t reduces to two terms (resp. proposition) u_1 and u_2, then there exists a term (resp. proposition) v such that u_1 reduces to v and u_2 reduces to v.*

Proposition 2.2 *In a finitely branching, terminating and confluent reduction system, a term (resp. a proposition) has exactly one irreducible form. And this irreducible form can be computed from the term (resp. the proposition).*

Proof. Termination yields existence. Confluence yields uniqueness as if u_1 and u_2 are irreducible forms of t, then $t \longrightarrow^* u_1$ and $t \longrightarrow^* u_2$. By confluence, there exists an expression v such that $u_1 \longrightarrow^* v$ and $u_2 \longrightarrow^* v$. As u_1 and u_2 are irreducible $u_1 = v = u_2$. To compute the irreducible form, it is sufficient to reduce the expression until an irreducible form is reached.

Proposition 2.3 *In a terminating and confluent reduction system, two terms (resp. two propositions) are congruent if and only if they have the same irreducible form.*

Proof. If the two expressions have the same irreducible form, then they are congruent. If they are congruent, so are their irreducible forms and these two irreducible forms reduce to a common expression. Hence they are equal.

Proposition 2.4 *In a terminating and confluent reduction system, the congruence is decidable.*

Proof. The congruence of two expressions can be checked by computing their irreducible forms and checking the identity of these irreducible forms.

2.2.3 Non confusion

Proposition 2.5 *In a confluent reduction system reducing terms to terms and atomic propositions to propositions, the congruence is non confusing.*

Proof. Consider two propositions A and A' such that $A \equiv A'$. Then either A or A' is atomic or both are non atomic. In this case, as the reduction relation is confluent, there exists a proposition A'' such that $A \longrightarrow^* A''$ and $A' \longrightarrow^* A''$.

As A and A' are non atomic, and the reduction rules reduce terms to terms and atomic propositions to propositions no reduction is performed at the root. Thus A'' is non atomic. If it is a conjunction, $A'' = B'' \wedge C''$ then $A = B \wedge C$, $A' = B' \wedge C'$, $B \longrightarrow^* B''$, $C \longrightarrow^* C''$, $B' \longrightarrow^* B''$, and $C' \longrightarrow^* C''$. Thus $B \equiv B'$ and $C \equiv C'$. We conclude in the same way if A'' is $\top$, $\bot$, a disjunction, an implication, a universal proposition, or an existential proposition.

2.3 Cuts in Deduction modulo theory

In Deduction modulo theory a cut is defined as in Predicate logic: it is a proof ending with an elimination rule whose main premise is proved by an introduction rule on the same symbol.

Proof reduction does not always terminate. Consider for instance the theory formed with the reduction rule $P \longrightarrow (P \Rightarrow Q)$ where P and Q are two proposition symbols. The proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{P \vdash P \Rightarrow Q}\,\text{axiom} \quad \dfrac{}{P \vdash P}\,\text{axiom}}{P \vdash Q}\,\Rightarrow\text{-elim}}{\vdash P \Rightarrow Q}\,\Rightarrow\text{-intro} \quad \dfrac{\dfrac{\dfrac{}{P \vdash P \Rightarrow Q}\,\text{axiom} \quad \dfrac{}{P \vdash P}\,\text{axiom}}{P \vdash Q}\,\Rightarrow\text{-elim}}{\vdash P}\,\Rightarrow\text{-intro}}{\vdash Q}\,\Rightarrow\text{-elim}$$

contains one cut because the conclusion $\vdash Q$ is proved using an elimination rule whose main premise $P \Rightarrow Q$ is proved using an introduction rule. This proof reduces to itself, hence its reduction does not terminate.

Exercise 2.1 *Prove that the sequent $\vdash Q$ has no constructive cut-free proof. Hint: what could be the last rule of such a proof?*

But, when proof reduction terminates, the cut-free proofs have the same properties than in Predicate logic.

Proposition 2.6 (Final rule) *If π is a proof of a proposition A that is (1) constructive, (2) cut-free, and (3) in a purely computational theory, then it ends with an introduction rule.*

Proof. As the proof π is a proof of the proposition A in a purely computational theory, it is a proof of the sequent $\vdash A$ modulo a congruence $\equiv$. By induction over the structure of this proof, we show that it ends with an introduction rule. As the proof is constructive, it ends either with an axiom rule, an elimination rule or an introduction rule. The last rule cannot be an axiom rule, because the left-hand part of the sequent is empty. If this last rule is an elimination, then the proof of the main premise of this elimination rule is constructive, cut-free and it proves a sequent whose left-hand part is empty. Hence, by induction hypothesis, it ends with an introduction and the proof is a cut, contradicting the fact that it is cut-free.

Proposition 2.7 (Disjunction property) *If a proposition $A \vee B$ has a proof that is (1.) constructive, (2.) cut-free, and (3.) in a purely computational theory, then there exists a proof of A or there exists a proof of B, and this proof is constructive, cut-free, and without any axioms.*

Proof. The proof ends with an introduction rule and, because the congruence is non confusing, this rule is a $\vee$-intro rule. The immediate sub-tree of this proof is a proof of A or a proof of B.

Proposition 2.8 (Witness property) *If a proposition $\exists x\ A$ has a proof that is (1.) constructive, (2.) cut-free, and (3.) in a purely computational theory, then there exists a term t and a proof of $(t/x)A$ that is constructive, cut-free, and without any axioms.*

Proof. The proof ends with an introduction rule and, because the congruence is non confusing, this rule is a $\exists$-intro rule. The immediate sub-tree of this proof is a proof of $(t/x)A$ for some term t.

Corollary 2.1 *All theories that are purely computational and where proof-reduction terminates have the disjunction and the witness property.*

Chapter 3

The Many-valued models

In proof-theory, as in other branches of logic, the notion of model is a central tool. In this chapter we generalize the usual notion of model, where propositions are interpreted in a two-element set $\{0,1\}$ needs to include models where propositions are interpreted in a larger set $\mathcal{B}$ of truth values. As we shall see, this generalization is useful to solve several problems: first the problem of building a single model to prove the independence of a proposition from a theory, then the adaptation of the notion of model to constructive logic and to Deduction modulo theory, finally to prove the termination of proof reduction for various theories (Theorem 9.1 and Corollaries 9.1 and 9.2).

We shall consider that the set $\mathcal{B}$ of truth values is equipped with a binary relation $\leq$, two distinguished elements $\tilde{\top}$ and $\tilde{\bot}$, three functions $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\Rightarrow}$ from $\mathcal{B} \times \mathcal{B}$ to $\mathcal{B}$, a function $\tilde{\forall}$ from a subset $\mathcal{A}$ of $\mathcal{P}^+(\mathcal{B})$ (the set of non-empty subsets of $\mathcal{B}$) to $\mathcal{B}$ and a function $\tilde{\exists}$ from a subset $\mathcal{E}$ of $\mathcal{P}^+(\mathcal{B})$ to $\mathcal{B}$. Such a structure will be called an *algebra*.

The usual notion of model valued in $\{0,1\}$ will be obtained as a particular case by taking $\mathcal{B} = \{0,1\}$, $\leq$ be the natural order on this set, $\tilde{\top} = 1$, $\tilde{\bot} = 0$, $\tilde{\wedge}$, $\tilde{\vee}$ and $\tilde{\Rightarrow}$ be the functions from $\{0,1\} \times \{0,1\}$ to $\{0,1\}$ defined by the following tables

$\tilde{\wedge}$	0	1
0	0	0
1	0	1

$\tilde{\vee}$	0	1
0	0	1
1	1	1

$\tilde{\Rightarrow}$	0	1
0	1	1
1	0	1

$\tilde{\forall}$ and $\tilde{\exists}$ be the functions from $\mathcal{P}^+(\{0,1\})$ to $\{0,1\}$ defined by the following tables

$\tilde{\forall}$	$\{0\}$	$\{0,1\}$	$\{1\}$
	0	0	1

$\tilde{\exists}$	$\{0\}$	$\{0,1\}$	$\{1\}$
	0	1	1

The notion of model valued in a Heyting algebra, that is usual in the theory of models of constructive logic, will be obtained as a particular case by taking the algebra $\mathcal{B}$ to be a Heyting algebra.

3.1 Model, valuation, interpretation, and validity

Definition 3.1 (Algebra) *An* algebra $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \mathcal{A}, \tilde{\forall}, \mathcal{E}, \tilde{\exists} \rangle$ *is a set* $\mathcal{B}$ *equipped with a binary relation* $\leq$ *on* $\mathcal{B}$*, two elements* $\tilde{\top}$ *and* $\tilde{\bot}$ *of* $\mathcal{B}$*, three functions* $\tilde{\wedge}$*,* $\tilde{\vee}$*, and* $\tilde{\Rightarrow}$ *from* $\mathcal{B} \times \mathcal{B}$ *to* $\mathcal{B}$*, a subset* $\mathcal{A}$ *of* $\mathcal{P}^+(\mathcal{B})$*, a function* $\tilde{\forall}$ *from* $\mathcal{A}$ *to* $\mathcal{B}$*, a subset* $\mathcal{E}$ *of* $\mathcal{P}^+(\mathcal{B})$*, and a function* $\tilde{\exists}$ *from* $\mathcal{E}$ *to* $\mathcal{B}$*.*

Definition 3.2 (Model) *Let* $\mathcal{L}$ *be a language in Predicate logic. A* model *for this language is a structure* $\mathcal{M}$ *formed with*

- *for each sort* s*, a non empty set* $\mathcal{M}_s$*,*
- *an algebra* $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \mathcal{A}, \tilde{\forall}, \mathcal{E}, \tilde{\exists}, \tilde{\Rightarrow} \rangle$*,*
- *for each function symbol* f *of arity* $\langle s_1, ..., s_n, s' \rangle$*, a function* $\hat{f}$ *from* $\mathcal{M}_{s_1} \times ... \times \mathcal{M}_{s_n}$ *to* $\mathcal{M}_{s'}$*,*
- *for each predicate symbol* P *of arity* $\langle s_1, ..., s_n \rangle$*, a function* $\hat{P}$ *from* $\mathcal{M}_{s_1} \times ... \times \mathcal{M}_{s_n}$ *to* $\mathcal{B}$*.*

Such a model is said to be valued *in the algebra* $\mathcal{B}$*.*

We want to define a function $[\![\ ]\!]$ that maps every term t of sort s, to an element $[\![t]\!]$ of $\mathcal{M}_s$ and every proposition A to an element $[\![A]\!]$ of $\mathcal{B}$. We moreover want this function to be a morphism, that is that $[\![f(t_1, ..., t_n)]\!] = \hat{f}([\![t_1]\!], ..., [\![t_n]\!])$, $[\![P(t_1, ..., t_n)]\!] = \hat{P}([\![t_1]\!], ..., [\![t_n]\!])$, $[\![A \wedge B]\!] = [\![A]\!] \tilde{\wedge} [\![B]\!]$, etc.

Such a morphism is completely defined by its image on the variables. This leads to the following definitions of the notions of *valuation* and *interpretation*.

Definition 3.3 (Valuation) *Let* $\mathcal{L}$ *be a language,* $(\mathcal{V}_s)_{s \in \mathcal{S}}$ *a family of disjoint sets of* variables*, and* $\mathcal{M}$ *a model of this language. A* valuation *is a function of finite domain mapping variables* $x_1, ..., x_n$ *of sorts* $s_1, ..., s_n$ *to elements* $a_1, ..., a_n$ *of* $\mathcal{M}_{s_1}$*, ...,* $\mathcal{M}_{s_n}$*.*

The valuation mapping the variable x_1 to a_1, ..., x_n to a_n is written $x_1 = a_1, ..., x_n = a_n$. If ϕ is a valuation, x a variable and a an element of $\mathcal{M}$ we write $(\phi, x = a)$ for the valuation that takes the same value as ϕ, everywhere except at x where it takes the value a.

Any valuation ϕ extends to a morphism $[\![\ ]\!]_\phi$ between the terms and the propositions of the language $\mathcal{L}$ whose free variables are in the domain of ϕ and the model $\mathcal{M}$.

Definition 3.4 (Interpretation) *Let* $\mathcal{L}$ *be a language,* $(\mathcal{V}_s)_{s \in \mathcal{S}}$ *a family of sets of variables,* $\mathcal{M}$ *a model of this language,* ϕ *a valuation and* t *a term whose free variables are in the domain of* ϕ*, the* interpretation *of the term* t *in the model* $\mathcal{M}$ *for the valuation* ϕ *is the element* $[\![t]\!]_\phi$ *of* $\mathcal{M}_s$ *defined as follows by induction on the structure of* t

- $[\![x]\!]_\phi = \phi(x)$
- $[\![f(t_1, ..., t_n)]\!]_\phi = \hat{f}([\![t_1]\!]_\phi, ..., [\![t_n]\!]_\phi)$

Let A be a proposition whose free variables are in the domain of ϕ, the interpretation *of the proposition A in the model $\mathcal{M}$ for the valuation ϕ is the element $[\![A]\!]_\phi$ of $\mathcal{B}$ defined as follows by induction on the structure of A*

- $[\![P(t_1, ..., t_n)]\!]_\phi = \hat{P}([\![t_1]\!]_\phi, ..., [\![t_n]\!]_\phi)$
- $[\![\top]\!]_\phi = \tilde{\top}$
- $[\![\bot]\!]_\phi = \tilde{\bot}$
- $[\![A \wedge B]\!]_\phi = [\![A]\!]_\phi \,\tilde{\wedge}\, [\![B]\!]_\phi$
- $[\![A \vee B]\!]_\phi = [\![A]\!]_\phi \,\tilde{\vee}\, [\![B]\!]_\phi$
- $[\![A \Rightarrow B]\!]_\phi = [\![A]\!]_\phi \,\tilde{\Rightarrow}\, [\![B]\!]_\phi$
- $[\![\forall x\ A]\!]_\phi = \tilde{\forall}\,\{[\![A]\!]_{\phi,x=a} \mid a \in \mathcal{M}_s\}$
- $[\![\exists x\ A]\!]_\phi = \tilde{\exists}\,\{[\![A]\!]_{\phi,x=a} \mid a \in \mathcal{M}_s\}$

Note that the function $[\![\ ]\!]_\phi$ is partial as the interpretation of the proposition $[\![\forall x\ A]\!]_\phi$ is defined only when the set $\{[\![A]\!]_{\phi,x=a} \mid a \in \mathcal{M}_s\}$ is an element of $\mathcal{A}$ and that of the proposition $[\![\exists x\ A]\!]_\phi$ is defined only when this set is an element of $\mathcal{E}$.

Definition 3.5 (Adequate) *A model is said to be* adequate *if all propositions of the language have an interpretation.*

Definition 3.6 (Full algebra) *An algebra is* full *if $\mathcal{A} = \mathcal{E} = \mathcal{P}^+(\mathcal{B})$, that is if $\tilde{\forall}$ and $\tilde{\exists}$ are always defined.*

Remark. A model whose algebra is full is adequate.

Proposition 3.1 (Substitution)

$$[\![(t/x)u]\!]_\phi = [\![u]\!]_{\phi,x=[\![t]\!]_\phi}$$

$$[\![(t/x)A]\!]_\phi = [\![A]\!]_{\phi,x=[\![t]\!]_\phi}$$

Proof. The first statement is proved by induction on the structure of u and the second by induction on the structure of A.

Definition 3.7 (Validity of a proposition) *Let $\mathcal{L}$ be a language, $(\mathcal{V}_s)_s$ be a family of sets of variables and $\mathcal{M}$ an adequate model of this language. A proposition A whose free variables are $x_1, ..., x_n$ is said to be* valid *in the model $\mathcal{M}$ if for all valuations ϕ whose domain contains $x_1, ..., x_n$, $[\![A]\!]_\phi \geq \tilde{\top}$.*

Definition 3.8 (Validity of a sequent) *A sequent $A_1, ..., A_n \vdash B$ is said to be* valid *in the model $\mathcal{M}$ if the proposition $(A_1 \wedge ... \wedge A_n) \Rightarrow B$ is, that is if for all valuations ϕ, $[\![A_1 \wedge ... \wedge A_n]\!]_\phi \,\tilde{\Rightarrow}\, [\![B]\!]_\phi \geq \tilde{\top}$.*

Definition 3.9 (Validity of an axiomatic theory) *An axiomatic theory $\mathcal{T}$ is said to be* valid *in a model if all its axioms are.*

Definition 3.10 (Validity of a congruence) *A congruence $\equiv$ is said to be* valid *in the model $\mathcal{M}$ if for all propositions A and B such that $A \equiv B$ and whose free variables are $x_1, ..., x_n$ and valuations ϕ whose domain contains $x_1, ..., x_n$, $[\![A]\!]_\phi = [\![B]\!]_\phi$.*

Definition 3.11 (Validity of a theory in Deduction modulo theory) *A theory $\mathcal{T}, \equiv$ is said to be* valid *in a model $\mathcal{M}$ if all the axioms of $\mathcal{T}$ and the congruence $\equiv$ are valid in $\mathcal{M}$.*

When a proposition—resp. a sequent, an axiomatic theory, a congruence, a theory in Deduction modulo theory—is valid in a model $\mathcal{M}$ we say also that $\mathcal{M}$ is *a model of* this proposition—resp. this sequent, this theory, this congruence, this theory.

Remark. In a model valued in the algebra $\{0, 1\}$, the condition $[\![A]\!]_\phi \geq \tilde{\top}$ boils down to $[\![A]\!]_\phi = 1$.

3.2 Pre-Boolean models

At this point, we can state and prove the usual soundness theorem for models valued in the set $\{0, 1\}$: if a sequent $\Gamma \vdash A$ has a proof, then it is valid in all the models valued in the algebra $\{0, 1\}$. The proof is a simple induction over the structure of the proof of $\Gamma \vdash A$.

From this theorem, we can deduce that if $\mathcal{T}$ is an axiomatic theory and A a proposition, then if A is provable in $\mathcal{T}$, it is valid in all models of $\mathcal{T}$. The contrapositive is that if there exists a model of $\mathcal{T}$ which is not a model of A, then A is not provable in $\mathcal{T}$. This result is the main tool to prove non provability results, that is results stating that some proposition is not provable in some theory. For instance, consider a language formed with two proposition symbols P and Q. It is easy to prove that, if we consider a single axiom P, then the proposition Q is not provable. Indeed, in the model valued in the algebra $\{0, 1\}$ and defined by $\hat{P} = 1$ and $\hat{Q} = 0$, the proposition P is valid, but Q is not.

In this theory, the proposition $\neg Q$ is not provable either, as in the model valued in the algebra $\{0, 1\}$ and defined by $\hat{P} = \hat{Q} = 1$, the proposition P is valid, but not the proposition $\neg Q$. Thus the proposition Q is *undetermined* in the theory P: neither Q nor its negation $\neg Q$ is provable in this theory.

The proof that Q is undetermined uses two distinct models and there is no way to show that the proposition Q is undetermined by using only one, as, in each model, either Q or $\neg Q$ is valid. Proving that the proposition Q is undetermined using a single model becomes possible if, instead of associating to each proposition a element of the set $\{0, 1\}$, we consider a larger set $\mathcal{B}$ of truth values. Consider, for instance, the powerset of any two-element set, such as the set $\{3, 4\}$. This set contains four elements $\varnothing, \{3\}, \{4\}, \{3, 4\}$. In this set, call 0 the empty set and 1 the set $\{3, 4\}$. Besides 0 and 1, the set $\mathcal{B}$ contains two intermediate truth values $\{3\}$ and $\{4\}$. Consider the model where $\hat{P} = 1$—that is $\{3, 4\}$—and $\hat{Q}$ is an intermediate value, for example $\{4\}$.

Then, as we shall see, in this model the interpretation of $\neg Q$ is $\{3\}$. Thus neither the interpretation of Q nor that of $\neg Q$ is 1 and neither Q not $\neg Q$ is valid in this model.

Note that this model aggregates the two models valued in the algebra $\{0, 1\}$ in a single one. Indeed, call $\mathcal{M}_3$ the first model above—defined by $\hat{P} = 1$ and $\hat{Q} = 0$—and $\mathcal{M}_4$ the second—defined by $\hat{P} = \hat{Q} = 1$. Then, the truth value of a proposition A in the new model is the set of indices i such that A is valid in $\mathcal{M}_i$.

This remark should lead to consider models where the set $\mathcal{B}$ is the powerset of some set E. But powersets are just examples of *Boolean algebras*. Thus, we can generalize further and consider models where the set $\mathcal{B}$ is a Boolean algebra.

Boolean algebras are sets equipped with an order relation that has a greatest lowerbound for the empty set ($\tilde{\top}$), binary greatest lowerbounds ($\tilde{\wedge}$), greatest lowerbounds for some sets ($\tilde{\forall}$), a least upperbound for the empty set ($\tilde{\bot}$), binary least upperbounds ($\tilde{\vee}$), least upperbounds for some sets ($\tilde{\exists}$), and a relative complement. In the case of the powerset of a set E, the order relation is the inclusion relation $\subseteq$, greatest lowerbounds are intersections, least upperbound are unions, and the relative complement of A and B is the set $(E \setminus A) \cup B$. When B is the empty set, the relative complement of A and $\varnothing$ is just the complement of A: $E \setminus A$.

But we can still generalize further. An order relation is a relation that is reflexive, antisymmetric, and transitive. Requiring this relation to be antisymmetric will complicate some proofs and we may consider pre-order relations instead, that is relations that are reflexive and transitive, but need not be antisymmetric. An intuitive explanation of why antisymmetry is not a natural condition is that the relation $\leq$ between propositions defined by $A \leq B$ if $A \Rightarrow B$ is provable is reflexive and transitive, but not antisymmetric. To make it antisymmetric we usually take the quotient of the set of propositions by the relation $\simeq$ defined by $A \simeq B$ if $A \Leftrightarrow B$ is provable. Thus, all the burden of dealing with quotients is avoided if we drop antisymmetry.

On the other hand, when working with pre-orders, we have to be careful that uniqueness of least upperbounds and greatest lowerbounds is lost. For instance if c_1 and c_2 are both a least upperbounds of a and b, then we have $c_1 \leq c_2$ and $c_2 \leq c_1$, but we cannot conclude that $c_1 = c_2$.

Thus, we shall consider models where the set $\mathcal{B}$ is a pre-Boolean algebra defined as follows.

Definition 3.12 (pre-Boolean algebra) *Let $\mathcal{B}$ be a set, $\leq$ be a relation on $\mathcal{B}$, $\mathcal{A}$ and $\mathcal{E}$ be subsets of $\mathcal{P}^+(\mathcal{B})$, $\tilde{\top}$ and $\tilde{\bot}$ be elements of $\mathcal{B}$, $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\Rightarrow}$ be functions from $\mathcal{B} \times \mathcal{B}$ to $\mathcal{B}$, $\tilde{\forall}$ be a function from $\mathcal{A}$ to $\mathcal{B}$ and $\tilde{\exists}$ be a function from $\mathcal{E}$ to $\mathcal{B}$. The structure $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \mathcal{A}, \tilde{\forall}, \mathcal{E}, \tilde{\exists} \rangle$ is said to be a* pre-Boolean algebra *if for all a, b, c in $\mathcal{B}$, A in $\mathcal{A}$ and E in $\mathcal{E}$,*
(the relation $\leq$ is an pre-order relation)

- $a \leq a$,
- *if $a \leq b$ and $b \leq c$ then $a \leq c$,*

($\tilde{\top}$ is a maximum element)

- $a \leq \tilde{\top}$,

($\tilde{\bot}$ is a minimum element)

- $\tilde{\bot} \leq a$,

($a \mathbin{\tilde{\wedge}} b$ is a greatest lower bound of a and b)

- $a \mathbin{\tilde{\wedge}} b \leq a$,
- $a \mathbin{\tilde{\wedge}} b \leq b$,
- *if* $c \leq a$ *and* $c \leq b$ *then* $c \leq a \mathbin{\tilde{\wedge}} b$,

($a \mathbin{\tilde{\vee}} b$ is a least upper bound of a and b)

- $a \leq a \mathbin{\tilde{\vee}} b$,
- $b \leq a \mathbin{\tilde{\vee}} b$,
- *if* $a \leq c$ *and* $b \leq c$ *then* $a \mathbin{\tilde{\vee}} b \leq c$,

($\tilde{\forall} A$ is a greatest lower bound of A)

- *if* $a \in A$ *then* $\tilde{\forall} A \leq a$,
- *if for all* a *in* A, $b \leq a$ *then* $b \leq \tilde{\forall} A$,

($\tilde{\exists} E$ is a least upper bound of E)

- *if* $a \in E$ *then* $a \leq \tilde{\exists} E$,
- *if for all* a *in* E, $a \leq b$ *then* $\tilde{\exists} E \leq b$,

($a \mathbin{\tilde{\Rightarrow}} b$ is a relative complement of a and b)

- $a \leq b \mathbin{\tilde{\Rightarrow}} c$ *if and only if* $a \mathbin{\tilde{\wedge}} b \leq c$,
- $\tilde{\top} \leq (a \mathbin{\tilde{\vee}} (a \mathbin{\tilde{\Rightarrow}} b))$.

Definition 3.13 (Boolean algebra) *A* Boolean algebra *is a pre-Boolean algebra whose relation $\leq$ is antisymmetric.*

Example 3.1 *The singleton $\{0\}$ trivially ordered and with the trivial operations is a pre-Boolean algebra, that is a Boolean algebra.*

Example 3.2 *The set $\{0, 1\}$ naturally ordered and equipped with the operations described in the introduction of this chapter is a pre-Boolean algebra, that is a Boolean algebra.*

Example 3.3 *Let E be any set, then the set $\mathcal{B} = \mathcal{P}(E)$ ordered by inclusion and equipped with the operations $\tilde{\top} = E$, $\tilde{\bot} = \varnothing$, $a \mathbin{\tilde{\wedge}} b = a \cap b$, $a \mathbin{\tilde{\vee}} b = a \cup b$, $\mathbin{\tilde{\Rightarrow}}(a, b) = (E \setminus a) \cup b$ $\mathcal{A} = \mathcal{E} = \mathcal{P}^+(\mathcal{B})$, $\tilde{\forall} A = \bigcap_{x \in A} x$, $\tilde{\exists} E = \bigcup_{x \in E} x$ is a pre-Boolean algebra, that is a Boolean algebra.*

Example 3.4 *Any set $\mathcal{B}$ pre-ordered with the relation $\leq$ defined by $a \leq b$ always and equipped with any operations is a pre-Boolean algebra. If $\mathcal{B}$ contains two elements or more, this pre-Boolean algebra is not a Boolean algebra.*

Definition 3.14 (Pre-Boolean Model) *A* pre-Boolean model *is a model where $\mathcal{B}$ is a pre-Boolean algebra.*

Remark that, as $[\![A]\!]_\phi \leq \tilde{\top}$, if the pre-Boolean algebra is Boolean, the condition $[\![A]\!]_\phi \geq \tilde{\top}$ is equivalent to $[\![A]\!]_\phi = \tilde{\top}$.

We can now state the soundness and completeness theorem for pre-Boolean models.

Theorem 3.1 (Soundness) *Let $\mathcal{T}$ be a theory in predicate logic. If the proposition A has a classical proof in $\mathcal{T}$, then it is valid in all pre-Boolean models of $\mathcal{T}$.*

Theorem 3.2 (Completeness) *Let $\mathcal{T}$ be a theory in predicate logic. If the proposition A is valid in all pre-Boolean models of $\mathcal{T}$, then it has a classical proof in $\mathcal{T}$.*

These theorems will be proved at the end of Section 3.3.

3.3 Pre-Heyting models

Like models valued in $\{0,1\}$, pre-Boolean models are models of the excluded-middle. The proposition $A \vee \neg A$, that is $A \vee (A \Rightarrow \bot)$ is always valid in such a model. Thus pre-Boolean models are complete for classical provability but not for constructive provability: some propositions of the form $A \vee \neg A$ have no constructive proofs, but they are valid in all pre-Boolean models.

But, moving from models valued in the algebra $\{0,1\}$ to models valued in a pre-Boolean algebra also shows the way to accommodate the notion of model to constructive proofs. All that is needed is to drop the last condition defining pre-Boolean algebras

$$\tilde{\top} \leq (a \,\tilde{\vee}\, (a \,\tilde{\Rightarrow}\, b))$$

We get this way, a notion of *pre-Heyting algebra*.

Definition 3.15 (pre-Heyting algebra) *Let $\mathcal{B}$ be a set, $\leq$ be a relation on $\mathcal{B}$, $\mathcal{A}$ and $\mathcal{E}$ be subsets of $\mathcal{P}^+(\mathcal{B})$, $\tilde{\top}$ and $\tilde{\bot}$ be elements of $\mathcal{B}$, $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\Rightarrow}$ be functions from $\mathcal{B} \times \mathcal{B}$ to $\mathcal{B}$, $\tilde{\forall}$ be a function from $\mathcal{A}$ to $\mathcal{B}$ and $\tilde{\exists}$ be a function from $\mathcal{E}$ to $\mathcal{B}$. The structure $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \mathcal{A}, \tilde{\forall}, \mathcal{E}, \tilde{\exists} \rangle$ is said to be a* pre-Heyting algebra *if for all a, b, c in $\mathcal{B}$, A in $\mathcal{A}$ and E in $\mathcal{E}$,*
(the relation $\leq$ is an pre-order relation)

- $a \leq a$,
- *if $a \leq b$ and $b \leq c$ then $a \leq c$,*

($\tilde{\top}$ is a maximum element)

- $a \leq \tilde{\top}$,

($\tilde{\bot}$ is a minimum element)

- $\tilde{\bot} \leq a$,

($a \mathbin{\tilde{\wedge}} b$ is a greatest lower bound of a and b)

- $a \mathbin{\tilde{\wedge}} b \leq a$,
- $a \mathbin{\tilde{\wedge}} b \leq b$,
- *if* $c \leq a$ *and* $c \leq b$ *then* $c \leq a \mathbin{\tilde{\wedge}} b$,

($a \mathbin{\tilde{\vee}} b$ is a least upper bound of a and b)

- $a \leq a \mathbin{\tilde{\vee}} b$,
- $b \leq a \mathbin{\tilde{\vee}} b$,
- *if* $a \leq c$ *and* $b \leq c$ *then* $a \mathbin{\tilde{\vee}} b \leq c$,

($\tilde{\forall} A$ in a greatest lower bound of A)

- *if* $a \in A$ *then* $\tilde{\forall} A \leq a$,
- *if for all* a *in* A, $b \leq a$ *then* $b \leq \tilde{\forall} A$,

($\tilde{\exists} E$ is a least upper bound of E)

- *if* $a \in E$ *then* $a \leq \tilde{\exists} E$,
- *if for all* a *in* E, $a \leq b$ *then* $\tilde{\exists} E \leq b$,

(and $a \mathbin{\tilde{\Rightarrow}} b$ is a weak relative complement of a and b)

- $a \leq b \mathbin{\tilde{\Rightarrow}} c$ *if and only if* $a \mathbin{\tilde{\wedge}} b \leq c$.

Definition 3.16 *A* Heyting algebra *is a pre-Heyting algebra whose relation $\leq$ is anti-symmetric.*

We can also give an alternative definition of the notion of pre-Boolean and Boolean algebra: A pre-Boolean algebra—resp. a Boolean algebra—is a pre-Heyting algebra—resp. a Heyting algebra—verifying the extra property

$$\tilde{\top} \leq (a \mathbin{\tilde{\vee}} (a \mathbin{\tilde{\Rightarrow}} b))$$

Example 3.5 *An example of a pre-Heyting algebra that is not a pre-Boolean algebra is given by the set of open sets of the real line—or of any topological space—ordered by inclusion and equipped with the following operations:* $\tilde{\top} = \mathbb{R}$, $\tilde{\bot} = \varnothing$, $a \mathbin{\tilde{\wedge}} b = a \cap b$, $a \mathbin{\tilde{\vee}} b = a \cup b$, $a \mathbin{\tilde{\Rightarrow}} b$ *is the interior of* $(\mathbb{R} \setminus a) \cup b$*—recall that the complement of an open set is not always open, for example if* $a = (-\infty, 0)$ *and* $b = \varnothing$*, then* $(\mathbb{R} \setminus a) \cup b$ *is* $[0, +\infty)$*—,* $\tilde{\forall} A$ *is the interior of* $\bigcap_{x \in A} x$*—recall that the intersection of an infinite set of open sets is not always open, for example* $\bigcap_{n \in \mathbb{N}} (0, 1+1/n) = (0, 1]$*—,* $\tilde{\exists} E = \bigcup_{x \in E} x$.

If $a = (-\infty, 0)$ *and* $b = \varnothing$*, then* $a \mathbin{\tilde{\vee}} (a \mathbin{\tilde{\Rightarrow}} b) = (-\infty, 0) \cup (0, +\infty)$*. This set does not contain* $\mathbb{R}$ *as a subset. Thus this pre-Heyting algebra is not a pre-Boolean algebra.*

Example 3.6 *Another example of pre-Heyting algebra that is not a pre-Boolean algebra is the set* $\{0, 1/2, 1\}$ *naturally ordered and equipped with the following operations* $\tilde{\top} = 1$, $\tilde{\bot} = 0$, $a \tilde{\wedge} b = glb(a, b)$, $a \tilde{\vee} b = lub(a, b)$, $\tilde{\forall} A = glb_{x \in A} x$ $\tilde{\exists} E = lub_{x \in E} x$, $a \tilde{\Rightarrow} b = b$ *if* $a > b$, *and* 1 *otherwise.*

It is routine to check that $a \leq (b \tilde{\Rightarrow} c)$ *if and only if* $(a \tilde{\wedge} b) \leq c$*—consider the three cases:* $b \leq c$, $b > c$ *and* $a \leq c$, *and finally* $b > c$ *and* $a > c$.

Then $1/2 \tilde{\vee} (1/2 \tilde{\Rightarrow} 0) = 1/2 \tilde{\vee} 0 = 1/2$. *Thus this pre-Heyting algebra is not a pre-Boolean algebra.*

Example 3.7 *An example of pre-Heyting algebra that is not a Heyting algebra is the set* $\{0, I, 1\}$ *pre-ordered with the relation*

	0	I	1
0	Y	N	N
I	Y	Y	Y
1	Y	Y	Y

and equipped with the operations $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\wedge}$, $\tilde{\vee}$, $\tilde{\forall}$, $\tilde{\exists}$, *and* $\tilde{\Rightarrow}$ *that are the same as in the algebra* $\{0, 1\}$, *except that their value on* I *is the same as their value on* 1*—for instance the table of the operation* $\tilde{\vee}$ *is*

	0	I	1
0	0	1	1
I	1	1	1
1	1	1	1

Remark as $I \leq 1$ *and* $1 \leq I$, *the relation* $\leq$ *is not antisymmetric and this algebra is not a Heyting algebra.*

Definition 3.17 (pre-Heyting Model) *A* pre-Heyting model *is a model where* $\mathcal{B}$ *is a pre-Heyting algebra.*

Remark that, if the pre-Heyting algebra is a Heyting algebra, the condition $[\![A]\!]_\phi \geq \tilde{\top}$ boils down to $[\![A]\!]_\phi = \tilde{\top}$.

We are now ready to prove the soundness and completeness theorems for pre-Heyting models.

Proposition 3.2 *In all pre-Heyting algebras,*

$$(a \tilde{\Rightarrow} b) \geq \tilde{\top} \text{ if and only if } a \leq b$$

thus the sequent $A_1, ..., A_n \vdash B$ *is valid in a model* $\mathcal{M}$ *if and only if for all valuations* ϕ, $[\![A_1 \wedge ... \wedge A_n]\!]_\phi \leq [\![B]\!]_\phi$.

Proof. If $\tilde{\top} \leq (a \tilde{\Rightarrow} b)$, then $a \leq \tilde{\top} \leq (a \tilde{\Rightarrow} b)$, and $a \tilde{\wedge} a \leq b$. As $a \leq a$ and $a \leq a$, we have $a \leq (a \tilde{\wedge} a)$. Thus, $a \leq (a \tilde{\wedge} a) \leq b$ and $a \leq b$.

Conversely, if $a \leq b$, then $(\tilde{\top} \tilde{\wedge} a) \leq a \leq b$ and $\tilde{\top} \leq a \tilde{\Rightarrow} b$.

Proposition 3.3 *If the sequent $\Gamma \vdash A$ has a constructive proof, then it is valid in all pre-Heyting models.*

Proof. By induction over proof structure, we give only one example. If the last rule is the introduction rule of the conjunction, the proof has the form

$$\frac{\dfrac{\pi_1}{H_1, ..., H_n \vdash B} \quad \dfrac{\pi_2}{H_1, ..., H_n \vdash C}}{H_1, ..., H_n \vdash B \wedge C} \wedge\text{-intro}$$

By induction hypothesis, the sequents $H_1, ..., H_n \vdash B$ and $H_1, ..., H_n \vdash C$ are valid in all models $\mathcal{M}$ of $\mathcal{T}$. Thus, for all model $\mathcal{M}$ and for all valuations ϕ, $[\![H_1 \wedge ... \wedge H_n]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![H_1 \wedge ... \wedge H_n]\!]_\phi \leq [\![C]\!]_\phi$. Hence $[\![H_1 \wedge ... \wedge H_n]\!]_\phi \leq [\![B]\!]_\phi \tilde{\wedge} [\![C]\!]_\phi$, that is $[\![H_1 \wedge ... \wedge H_n]\!]_\phi \leq [\![B \wedge C]\!]_\phi$, and the sequent $H_1, ..., H_n \vdash B \wedge C$ is valid in $\mathcal{M}$.

Theorem 3.3 (Soundness) *Let $\mathcal{T}$ be an axiomatic theory. If the proposition A has a constructive proof in $\mathcal{T}$, then it is valid in all pre-Heyting models of $\mathcal{T}$.*

Proof. If A as a constructive proof in $\mathcal{T}$, there exists a finite subset $\{H_1, ..., H_p\}$ of $\mathcal{T}$ such that the sequent $H_1, ..., H_p \vdash A$ has a constructive proof. Thus, for all models $\mathcal{M}$ and valuations ϕ, $[\![H_1 \wedge ... \wedge H_n]\!]_\phi \leq [\![A]\!]_\phi$.

Let $\mathcal{M}$ be a model of $\mathcal{T}$ and ϕ be any valuation, we have $\tilde{\top} \leq [\![H_1]\!]_\phi$, ..., $\tilde{\top} \leq [\![H_p]\!]_\phi$ thus $\tilde{\top} \leq [\![H_1]\!]_\phi \tilde{\wedge} ... \tilde{\wedge} [\![H_p]\!]_\phi$ that is $\tilde{\top} \leq [\![H_1 \wedge ... \wedge H_p]\!]_\phi$. Therefore $\tilde{\top} \leq [\![A]\!]_\phi$ and the proposition A is valid in $\mathcal{M}$.

We now want to prove the converse, that is that if a proposition is valid in all pre-Heyting models of a theory $\mathcal{T}$, then it has a constructive proof in $\mathcal{T}$. Thanks to our many generalizations of the algebra $\{0, 1\}$, this theorem has a simple proof, that proceeds by building a single model where valid propositions are exactly those that have a constructive proof in $\mathcal{T}$.

Definition 3.18 (The Lindenbaum model) *Let $\mathcal{T}$ be an axiomatic theory in a language $\mathcal{L}$. Let S be a set containing an infinite number of constants of each sort and $\mathcal{L}' = \mathcal{L} \cup S$. Let $\mathcal{M}_s$ be the set of closed terms of $\mathcal{L}'$ of sort s and $\mathcal{B}$ be the set of closed propositions of $\mathcal{L}'$.*

We define the relation $\leq$ by $A \leq B$ if and only if $A \Rightarrow B$ has a constructive proof in $\mathcal{T}$.

Let $\mathcal{A} = \mathcal{E}$ be the set of subsets of $\mathcal{B}$ of the form $\{(t/x)A \mid t \in \mathcal{M}_s\}$ for some A. Note that, in this case, the proposition A is unique.

The operations $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\Rightarrow}$, are $\top$, $\bot$, $\wedge$, $\vee$, and $\Rightarrow$. The operations $\tilde{\forall}$ and $\tilde{\exists}$ are defined as follows

- $\tilde{\forall}\,\{(t/x)A \mid t \in \mathcal{M}_s\} = (\forall x\ A)$,
- $\tilde{\exists}\,\{(t/x)A \mid t \in \mathcal{M}_s\} = (\exists x\ A)$.

If f is a function symbol, we let $\hat{f}$ be the function mapping $t_1, ..., t_n$ to $f(t_1, ..., t_n)$. If P is a predicate symbol, we let $\hat{P}$ be the function mapping $t_1, ..., t_n$ to $P(t_1, ..., t_n)$.

Proposition 3.4 *Let A be a proposition and ϕ be a valuation mapping the free variables of A to elements of $\mathcal{M}$. Note that ϕ is also a substitution and that ϕA is a closed proposition. Then*

$$[\![A]\!]_\phi = \phi A$$

Proof. By induction over the structure of A. We consider only the case where $A = \forall x\ B$. We have $[\![\forall x\ B]\!]_\phi = \tilde{\forall}\ \{[\![B]\!]_{\phi,x=t} \mid t \in \mathcal{M}_s\} = \tilde{\forall}\ \{[\![(t/x)B]\!]_\phi \mid t \in \mathcal{M}_s\} = \tilde{\forall}\ \{\phi((t/x)B) \mid t \in \mathcal{M}_s\} = \tilde{\forall}\ \{(t/x)(\phi B) \mid t \in \mathcal{M}_s\} = \forall x\ (\phi B) = \phi(\forall x\ B)$.

Proposition 3.5 *The Lindenbaum model of $\mathcal{T}$ is adequate.*

Proof. By construction, all the sets of the form $\{[\![A]\!]_{\phi,x=t} \mid t \in \mathcal{M}_s\} = \{(t/x)A \mid t \in \mathcal{M}_s\}$ are in $\mathcal{A}$ and in $\mathcal{E}$.

Proposition 3.6 *The algebra of the Lindenbaum model of $\mathcal{T}$ is a pre-Heyting algebra.*

Proof. The relation $\leq$ is reflexive because the proposition $A \Rightarrow A$ always has a constructive proof and it is transitive because if $A \Rightarrow B$ and $B \Rightarrow C$ have constructive proofs, then so does the proposition $A \Rightarrow C$ (but note that it needs not be antisymmetric).

The element $\top$ is a maximum because $A \Rightarrow \top$ always has a constructive proof, the element $\bot$ is a minimum because $\bot \Rightarrow A$ always has a constructive proof.

The conjunction is a greatest lowerbound because $(A \wedge B) \Rightarrow A$ always has a constructive proof, $(A \wedge B) \Rightarrow B$ always has a constructive proof, and if $C \Rightarrow A$ and $C \Rightarrow B$ have constructive proofs, then so does $C \Rightarrow (A \wedge B)$.

The disjunction is a least upperbound because $A \Rightarrow (A \vee B)$ always has a constructive proof, $B \Rightarrow (A \vee B)$ always has a constructive proof and if $A \Rightarrow C$ and $B \Rightarrow C$ have constructive proofs, then so does $(A \vee B) \Rightarrow C$.

The proposition $\tilde{\forall}\ \{(t/x)A \mid t \in \mathcal{M}_s\} = \forall x\ A$ is a greatest lowerbound of $\{(t/x)A \mid t \in \mathcal{M}_s\}$ because for all terms t the proposition $(\forall x\ A) \Rightarrow (t/x)A$ has a constructive proof and if for all terms t, $B \Rightarrow (t/x)A$ has a constructive proof then, as there is an infinite number of constants, there exists a constant c that does not occur in B and, as the proposition $B \Rightarrow (c/x)A$ has a constructive proof, then so does the proposition $B \Rightarrow \forall x\ A$.

The proposition $\tilde{\exists}\ \{(t/x)A \mid t \in \mathcal{M}_s\} = \exists x\ A$ is a least upperbound of $\{(t/x)A \mid t \in \mathcal{M}_s\}$ because for all terms t the proposition $((t/x)A) \Rightarrow \exists x\ A$ has a constructive proof and if for all terms t, $((t/x)A) \Rightarrow B$ has a constructive proof then, as there is an infinite number of constants, there exists a constant c that does not occur in B and, as the proposition $((c/x)A) \Rightarrow B$ has a constructive proof, so does the proposition $(\exists x\ A) \Rightarrow B$.

Finally, the implication is a weak relative complement, because if $A \Rightarrow (B \Rightarrow C)$ has a constructive proof, then so does the proposition $(A \wedge B) \Rightarrow C$ and if $(A \wedge B) \Rightarrow C$ has a constructive proof, then so does the proposition $A \Rightarrow (B \Rightarrow C)$.

Proposition 3.7 *The Lindenbaum model of $\mathcal{T}$ is a model of $\mathcal{T}$.*

Proof. If A is a proposition of $\mathcal{T}$, then $\top \Rightarrow A$ has a constructive proof in $\mathcal{T}$ and thus for all valuations ϕ, $[\![A]\!]_\phi = \phi A = A \geq \top$.

Proposition 3.8 *If a proposition is valid in the Lindenbaum model of $\mathcal{T}$ then it has a constructive proof in $\mathcal{T}$.*

Proof. Let $x_1, ..., x_n$ be the free variables of A and $c_1, ..., c_n$ be constants of S of the same sort. If A is valid in the Lindenbaum model of $\mathcal{T}$, then $[\![A]\!]_{x_1=c_1,...,x_n=c_n} \geq \tilde{\top}$, $\top \Rightarrow (c_1/x_1, ..., c_n/x_n)A$ has a constructive proof in $\mathcal{T}$, thus $(c_1/x_1, ..., c_n/x_n)A$ has a constructive proof in $\mathcal{T}$ and A has a constructive proof in $\mathcal{T}$.

Theorem 3.4 (Completeness) *If a proposition is valid in all pre-Heyting models of $\mathcal{T}$, then it has a constructive proof in $\mathcal{T}$.*

Proof. If a proposition is valid in all pre-Heyting models of $\mathcal{T}$, it is valid in the Lindenbaum model of $\mathcal{T}$.

Remark. The proofs of Theorem 3.1 is the same as that of the Theorem 3.3, except that we have one more rule to check: the excluded-middle, and to check that all propositions of the form $A \vee \neg A$, that is $A \vee (A \Rightarrow \bot)$, are valid we use the fact that in a pre-Boolean algebra

$$\tilde{\top} \leq (a \,\tilde{\vee}\, (a \,\tilde{\Rightarrow}\, b))$$

The proofs of the theorem 3.2 is the same as that of the Theorem 3.4, except that the relation $\leq$ of the algebra of the Lindenbaum model defined by $A \leq B$ if $A \Rightarrow B$ has a classical proof in $\mathcal{T}$. Then, we check that this algebra is not only a pre-Heyting algebra but also a pre-Boolean algebra, that is that it verifies the property

$$\tilde{\top} \leq (a \,\tilde{\vee}\, (a \,\tilde{\Rightarrow}\, b))$$

and this is because the proposition $A \vee (A \Rightarrow B)$ always has a classical proof.

3.4 Consistency

Definition 3.19 (Consistency) *A theory is* consistent *if there exists a proposition that is not provable in this theory.*

Proposition 3.9 *A theory is consistent if and only if the proposition $\bot$ is not provable in this theory.*

Proof. If the proposition $\bot$ is not provable, then the theory is consistent. If it is, then, by the rule $\bot$-elim, all propositions are.

In the Soundness and Completeness theorems, we have not assumed that the theory $\mathcal{T}$ was consistent. Indeed, even non consistent theories have pre-Heyting models. For instance, the singleton $\{0\}$ equipped with the trivial operations is a pre-Heyting algebra and every proposition, even $\bot$, is valid in any model whose pre-Heyting algebra is this algebra.

But consistent axiomatic theories have models whose pre-Heyting algebra is *non trivial*.

Definition 3.20 *A pre-Heyting algebra is trivial if $a \leq b$ always.*

Proposition 3.10 *In a non trivial pre-Heyting algebra,* $\top \not\leq \bot$.

Proof. Assume $\top \leq \bot$. Then, for all a and b, we have $a \leq \top \leq \bot \leq b$, and, by transitivity, $a \leq b$.

Theorem 3.5 (Consistency) *An axiomatic theory* $\mathcal{T}$ *is consistent if and only if it has a model whose pre-Heyting algebra is non trivial.*

Proof. If $\mathcal{T}$ is consistent, then the proposition $\bot$ is not provable in $\mathcal{T}$. Thus, by Proposition 3.8, this proposition is not valid in the Lindenbaum model of $\mathcal{T}$ and in the algebra of this model $\bot \not\geq \top$. Thus, by Proposition 3.7, the Lindenbaum model of $\mathcal{T}$ is a model of $\mathcal{T}$ and its algebra is non trivial.

Conversely, assume that theory $\mathcal{T}$ has a model $\mathcal{M}$ whose pre-Heyting algebra is non trivial. Then, by Proposition 3.10, in this pre-Heyting algebra $\bot \not\geq \top$, thus $\bot$ is not valid in $\mathcal{M}$, $\bot$ is not valid in all models of $\mathcal{T}$ and, by the Soundness theorem (Theorem 3.3), $\bot$ is not provable in $\mathcal{T}$.

3.5 Models of Deduction modulo theory

In this section, we extend the Soundness, the Completeness and the Consistency theorem to Deduction modulo theory. The intuitive idea is to replace terms by classes of congruent terms and propositions by classes of congruent propositions.

The only difficulty is in the definition of the function $\tilde{\forall}$ in the algebra of the Lindenbaum model: $\tilde{\forall}\,\{(t/x)A \mid t \in \mathcal{M}_s\} = (\forall x\ A)$, indeed, this definition presupposes that the proposition A is uniquely defined by the set of its instances $\{(t/x)A \mid t \in \mathcal{M}_s\}$. This is the case in Predicate logic, but not in Deduction modulo theory. For instance, if the congruence $\equiv$ is defined by the reduction rule $f(f(x)) \longrightarrow x$, the propositions $P(x)$ and $P(f(x))$ have the same instances: the instance t in one proposition corresponds to the instance $f(t)$ in the other. But the propositions $\forall x\ P(x)$ and $\forall x\ P(f(x))$ are not congruent. This will lead us to extend the relation $\equiv$ to a relation $\sim$ such that $\forall x\ P(x) \sim \forall x\ P(f(x))$.

Proposition 3.11 *If the sequent* $A_1, ..., A_n \vdash B$ *has a constructive proof modulo* $\equiv$*, then it is valid in all pre-Heyting models of* $\equiv$*.*

Proof. By induction over proof structure, using the fact that the model is a model of the congruence to justify the replacement of a proposition by a congruent one in each rule.

Theorem 3.6 (Soundness) *Let* $\mathcal{T}, \equiv$ *be a theory in Deduction modulo theory. If a proposition has a constructive proof in* $\mathcal{T}, \equiv$*, then it is valid in all pre-Heyting models of* $\mathcal{T}, \equiv$*.*

Proof. Same as that of Theorem 3.3.

We now want to prove the converse, that is that if a proposition is valid in all pre-Heyting models of $\mathcal{T}, \equiv$, then it has a constructive proof in $\mathcal{T}, \equiv$. Again, we build a single model where valid propositions are exactly those that have a constructive proof in $\mathcal{T}$.

Definition 3.21 (The relation $\sim$) *Let $\mathcal{T}, \equiv$ be a theory in a language $\mathcal{L}$. Let S be a set containing an infinite number of constants of each sort and $\mathcal{L}' = \mathcal{L} \cup S$. The relation $\sim$ is inductively defined on propositions of $\mathcal{L}'$ as the smallest congruence such that*

- *if $A \equiv B$ then $A \sim B$,*
- *if $A \sim B$ and $A' \sim B'$ then $(A \wedge A') \sim (B \wedge B')$, $(A \vee A') \sim (B \vee B')$, and $(A \Rightarrow A') \sim (B \Rightarrow B')$,*
- *and if for each term t there exist a term u such that $(t/x)A \sim (u/x)B$ and for each term u there exist a term t such that $(u/x)B \sim (t/x)A$ then $\forall x\ A \sim \forall x\ B$ and $\exists x\ A \sim \exists x\ B$.*

Proposition 3.12 *If $t \equiv u$ and $A \sim B$ then $(t/x)A \sim (u/x)B$.*

Proof. By induction on the structure of the derivation of $A \sim B$.

Proposition 3.13 *If $A \sim B$ then $A \Leftrightarrow B$ is provable modulo $\equiv$.*

Proof. By induction on the structure of the derivation of $A \sim B$. If $A = \forall x\ A'$ and $B = \forall x\ B'$ and for each term t there exist a term u such that $(t/x)A' \sim (u/x)B'$ and for each term u there exist a term t such that $(u/x)B' \sim (t/x)A'$ then let c be constants of S occurring neither in A' nor in B'. We have to prove the sequent $\forall x\ A' \vdash (c/x)B'$. Let t be a term such that $(t/x)A' \sim (c/x)B'$, by induction hypothesis we get $(t/x)A' \Leftrightarrow (c/x)B'$, and as we have $\forall x\ A'$ we can deduce $(t/x)A'$, and thus $(c/x)B'$.

Definition 3.22 (The Lindenbaum model) *Let $\mathcal{T}, \equiv$ be a theory in a language $\mathcal{L}$. Let S be a set containing an infinite number of constants of each sort and $\mathcal{L}' = \mathcal{L} \cup S$. Let $\mathcal{M}_s$ be the set of $\equiv$-classes of closed terms of $\mathcal{L}'$ of sort s and $\mathcal{B}$ be the set of $\sim$-classes of closed propositions of $\mathcal{L}'$.*

We define the relation $\leq$ by $A \leq B$ if and only if $C \Rightarrow D$ has a constructive proof in $\mathcal{T}, \equiv$, where C is an element of A and D an element of B. Note that, by Proposition 3.13, this definition is independent of the choice of C and D.

Let $\mathcal{A} = \mathcal{E}$ be the set of subsets of $\mathcal{B}$ of the form $\{(t/x)A/\sim \mid t \in \mathcal{M}_s\}$ for some A.

The operations $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\Rightarrow}$, $\tilde{\wedge}$ and $\tilde{\vee}$ are $\top$, $\bot$, $\Rightarrow$, $\wedge$ and $\vee$ extended to $\sim$-classes. To define the operations $\tilde{\forall}$ and $\tilde{\exists}$, we choose for each element a of $\mathcal{A}$ and $\mathcal{E}$ a proposition A such that $a = \{(t/x)A/\sim \mid t \in \mathcal{M}_s\}$ and we let

- $\tilde{\forall}\ a = (\forall x\ A)/\sim$,
- $\tilde{\exists}\ a = (\exists x\ A)/\sim$.

Note that, by construction of $\sim$ and because of Proposition 3.12, the elements $(\forall x\ A)/\sim$ and $(\exists x\ A)/\sim$ are independent of the choice of A (while this would not be the case with the relation $\equiv$).

If f is a function symbol, we let $\hat{f}$ be the function mapping the classes of $t_1, ..., t_n$ to that of $f(t_1, ..., t_n)$.

If P is a predicate symbol, we let $\hat{P}$ be the function mapping the classes of $t_1, ..., t_n$ to that of $P(t_1, ..., t_n)$.

Proposition 3.14 *Let A be a proposition and ϕ be a valuation mapping the free variables of A to elements of $\mathcal{M}$. Note that ϕ is also a substitution and that ϕA is a closed proposition. Then $[\![A]\!]_\phi = \phi A/\sim$.*

Proof. By induction over the structure of A. We consider only the case where $A = \forall x\ B$. We have $[\![\forall x\ B]\!]_\phi = \tilde{\forall}\ \{[\![B]\!]_{\phi,x=a} \mid a \in \mathcal{M}_s\}$. By induction hypothesis, $[\![\forall x\ B]\!]_\phi = \tilde{\forall}\ \{(a/x)\phi B/\sim \mid a \in \mathcal{M}_s\} = (\forall x\ \phi B)/\sim = \phi(\forall x\ B)/\sim$.

Proposition 3.15 *The Lindenbaum model of $\mathcal{T}, \equiv$ is adequate.*

Proof. By construction, all the sets of the form $\{[\![A]\!]_{x=t} \mid t \in \mathcal{M}_s\} = \{(t/x)A/\sim \mid t \in \mathcal{M}_s\}$ are in $\mathcal{A}$ and in $\mathcal{E}$.

Proposition 3.16 *The algebra of the Lindenbaum model of $\mathcal{T}, \equiv$ is a pre-Heyting algebra.*

Proof. The relation $\leq$ is reflexive because the proposition $A \Rightarrow A$ always has a constructive proof and it is transitive because if $A \Rightarrow B$ and $B \Rightarrow C$ have constructive proofs, then so does the proposition $A \Rightarrow C$.

The element $\top$ is a maximum because $A \Rightarrow \top$ always has a constructive proof, the element $\bot$ is a minimum because $\bot \Rightarrow A$ always has a constructive proof.

The conjunction is a greatest lowerbound because $(A \wedge B) \Rightarrow A$ always has a constructive proof, $(A \wedge B) \Rightarrow B$ always has a constructive proof, and if $C \Rightarrow A$ and $C \Rightarrow B$ have constructive proofs, then so does $C \Rightarrow (A \wedge B)$.

The disjunction is a least upperbound because $A \Rightarrow (A \vee B)$ always has a constructive proof, $B \Rightarrow (A \vee B)$ always has a constructive proof and if $A \Rightarrow C$ and $B \Rightarrow C$ have constructive proofs, then so does $(A \vee B) \Rightarrow C$.

The proposition $\tilde{\forall}\ \{(t/x)A/\sim \mid t \in \mathcal{M}_s\} = \forall x\ A/\sim$ is a greatest lowerbound of $\{(t/x)A/\sim \mid t \in \mathcal{M}_s\}$ because for all terms t the proposition $(\forall x\ A) \Rightarrow (t/x)A$ has a constructive proof and if for all terms t, $B \Rightarrow (t/x)A$ has a constructive proof then, as there is an infinite number of constants, there exists a constant c that does not occur in B and, as the proposition $B \Rightarrow (c/x)A$ has a constructive proof, then so does the proposition $B \Rightarrow \forall x\ A$.

The proposition $\tilde{\exists}\ \{(t/x)A/\sim \mid t \in \mathcal{M}_s\} = \exists x\ A/\sim$ is a least upperbound of $\{(t/x)A/\sim \mid t \in \mathcal{M}_s\}$ because for all terms t the proposition $((t/x)A) \Rightarrow \exists x\ A$ has a constructive proof and if for all terms t, $((t/x)A) \Rightarrow B$ has a constructive proof then, as there is an infinite number of constants, there exists a constant c that does not occur in B and, as the proposition $((c/x)A) \Rightarrow B$ has a constructive proof, so does the proposition $(\exists x\ A) \Rightarrow B$.

Finally, the implication is a weak relative complement, because if $A \Rightarrow (B \Rightarrow C)$ has a constructive proof, then so does the proposition $(A \wedge B) \Rightarrow C$ and if $(A \wedge B) \Rightarrow C$ has a constructive proof, then so does the proposition $A \Rightarrow (B \Rightarrow C)$.

Proposition 3.17 *The Lindenbaum model of $\mathcal{T}, \equiv$ is a model of $\mathcal{T}, \equiv$.*

Proof. If A is a proposition of $\mathcal{T}$, then $\top \Rightarrow A$ has a constructive proof in $\mathcal{T}, \equiv$ and thus for all valuations ϕ, $[\![A]\!]_\phi = \phi A = A \geq \top$. If $A \equiv B$ then $A \sim B$, thus $\phi A \sim \phi B$ $\phi A/\sim = \phi B/\sim$ and $[\![A]\!]_\phi = [\![B]\!]_\phi$.

Proposition 3.18 *If a proposition is valid in the Lindenbaum model of $\mathcal{T}, \equiv$ then it has a constructive proof in $\mathcal{T}, \equiv$.*

Proof. Let $x_1, ..., x_n$ be the free variables of A and $c_1, ..., c_n$ be constants of S of the same sort. If A is valid in the Lindenbaum model of $\mathcal{T}, \equiv$, then $[\![A]\!]_{x_1=c_1,...,x_n=c_n} \geq \tilde{\top}, \top \Rightarrow (c_1/x_1, ..., c_n/x_n)A$ has a constructive proof in $\mathcal{T}, \equiv$, thus $(c_1/x_1, ..., c_n/x_n)A$ has a constructive proof in $\mathcal{T}, \equiv$ and A has a constructive proof in $\mathcal{T}, \equiv$.

Theorem 3.7 (Completeness) *If a proposition is valid in all pre-Heyting models of $\mathcal{T}, \equiv$, then it has a constructive proof in $\mathcal{T}, \equiv$.*

Proof. If a proposition is valid in all pre-Heyting models of $\mathcal{T}, \equiv$, it is valid in the Lindenbaum model of $\mathcal{T}, \equiv$.

Theorem 3.8 (Consistency) *A theory $\mathcal{T}, \equiv$ is consistent if and only if it has a model whose pre-Heyting algebra is non trivial.*

Proof. If $\mathcal{T}, \equiv$ is consistent, then the proposition $\bot$ is not provable in $\mathcal{T}, \equiv$. Thus, by Proposition 3.18, this proposition is not valid in the Lindenbaum model of $\mathcal{T}, \equiv$ and in the algebra of this model $\bot \not\geq \top$. Thus, by Proposition 3.17, the Lindenbaum model of $\mathcal{T}, \equiv$ is a model of $\mathcal{T}, \equiv$ and its algebra is non trivial.

Conversely, assume that theory $\mathcal{T}, \equiv$ has a model $\mathcal{M}$ whose pre-Heyting algebra is non trivial. Then, by Proposition 3.10, in this pre-Heyting algebra $\bot \not\geq \top$, thus $\bot$ is not valid in $\mathcal{M}$, $\bot$ is not valid in all models of $\mathcal{T}, \equiv$ and, by the Soundness theorem (Theorem 3.6), $\bot$ is not provable in $\mathcal{T}, \equiv$.

Remark. Let $\mathcal{T}, \equiv$ be a theory and $\mathcal{M}$ be a model of this theory. If the proposition $A \Leftrightarrow B$ is provable in the theory $\mathcal{T}, \equiv$, then for all valuations ϕ, $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![B]\!]_\phi \leq [\![A]\!]_\phi$. In contrast, if $A \equiv B$, then for all valuations ϕ, $[\![A]\!]_\phi = [\![B]\!]_\phi$. In a Heyting algebra these conditions are the equivalent because, by antisymmetry, $[\![A]\!]_\phi \leq [\![B]\!]_\phi$ and $[\![B]\!]_\phi \leq [\![A]\!]_\phi$ implies $[\![A]\!]_\phi = [\![B]\!]_\phi$, but not in a pre-Heyting algebra. Thus, using pre-Heyting algebras instead of Heyting algebras allows to distinguish in a model, the strong equivalence $\equiv$ from the weaker one $\Leftrightarrow$. For instance the propositions $4 = 4$ and $2+2 = 4$ are strongly equivalent, while Fermat's little theorem and Fermat's last theorem are only weakly equivalent. Distinguishing strong and weak equivalence in a model is one of the motivations for dropping antisymmetry.

3.6 Super-consistency

Consider the purely computational theory defined by the reduction rule $P \longrightarrow (Q \Rightarrow Q)$. To prove the consistency of this theory, we can use Theorem 3.8 and build a model where $\hat{P} = \hat{Q} \mathrel{\tilde{\Rightarrow}} \hat{Q}$, as it is easy to prove that in such a model, if $A \equiv B$ then $[\![A]\!]_\phi = [\![B]\!]_\phi$. A possibility is to consider the pre-Heyting algebra $\{0, 1\}$ where $\tilde{\Rightarrow}$ is

$\tilde{\Rightarrow}$	0	1
0	1	1
1	0	1

and then construct the model valued in the algebra $\{0,1\}$ defined by $\hat{Q} = 1$ and $\hat{P} = 1 \tilde{\Rightarrow} 1 = 1$.

But it is clear that no property of the algebra $\{0,1\}$ is used in this construction. If $\mathcal{B}$ is any pre-Heyting algebra, then by taking $\hat{Q} = \tilde{\top}$ and $\hat{P} = \tilde{\top} \tilde{\Rightarrow} \tilde{\top}$, we obtain a model of this theory valued in $\mathcal{B}$. Thus, not only this theory has a model valued in the algebra $\{0,1\}$, but also a model valued in any pre-Heyting algebra $\mathcal{B}$.

This property of having a model valued in any pre-Heyting algebras $\mathcal{B}$ will be useful when proving the termination of proof reduction. In fact, we will need a slightly weaker notion: a theory will be called *super-consistent* if it has a model valued in any *full*, *ordered*, and *complete* pre-Heyting algebra.

Definition 3.23 (Ordered pre-Heyting algebra) *An* ordered pre-Heyting algebra *is a pre-Heyting algebra equipped with an extra binary relation $\sqsubseteq$ on $\mathcal{B}$ such that*

- *$\sqsubseteq$ is an order relation,*
- *$\tilde{\wedge}$, $\tilde{\vee}$, $\tilde{\forall}$, and $\tilde{\exists}$ are monotone, $\tilde{\Rightarrow}$ is left anti-monotone and right monotone.*

Definition 3.24 (Complete ordered pre-Heyting algebra) *A ordered pre-Heyting algebra is said to be* complete *if every subset of $\mathcal{B}$ has a greatest lower bound for $\sqsubseteq$.*

Definition 3.25 (Super-consistent) *A theory $\mathcal{T}, \equiv$ in Deduction modulo theory is* super-consistent *if it has a model valued in any full, ordered, and complete pre-Heyting algebra.*

Requiring the pre-Heyting algebra to be full is just a matter of simplicity: it makes the interpretation function $[\![.]\!]_\phi$ total. Requiring the pre-Heyting algebra to be ordered and complete is more important, as it will permit us to build models for some theories as fixed-points, using the fact that a monotone function on any complete order has a fixed-point.

Example 3.8 *The theory defined by the reduction rule*

$$P \longrightarrow (Q \Rightarrow Q)$$

is super-consistent. Indeed, consider any full, ordered, and complete pre-Heyting algebra $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \mathcal{A}, \tilde{\forall}, \mathcal{E}, \tilde{\exists} \rangle$. Then, the model $\mathcal{M}$ formed with any set $\mathcal{M}$, the algebra $\mathcal{B}$, and the functions $\hat{Q} = \tilde{\top}$ and $\hat{P} = \tilde{\top} \tilde{\Rightarrow} \tilde{\top}$ is a model of this theory valued in $\mathcal{B}$.

Chapter 4

Arithmetic

Arithmetic is the theory of natural numbers. Examples of propositions that can be proved in arithmetic are $\exists y\ (4 = 2 \times y)$, $\forall x \exists y\ (x = 2 \times y \vee x = 2 \times y + 1)$, etc. The terms 2, 4, etc. are neither constants, nor terms written in binary or in decimal notation, they are terms written in unary notation: with a constant 0, *zero*, and a unary function symbol S, *successor*, thus the number 1 is written $S(0)$, the number 2 is written $S(S(0))$, etc.

This theory can be used with classical logic or with constructive logic. In the first case, it is called *Peano arithmetic* (PA) and in the second *Heyting arithmetic* (HA).

Both Peano arithmetic and Heyting arithmetic have several formulations: with or without a sort κ for classes and with or without a predicate symbol N for natural numbers. In this chapter, we focus on the presentation $\text{HA}^{\kappa N}$ that uses both a sort for classes and a predicate symbol N for natural numbers. As we shall see, this formulation $\text{HA}^{\kappa N}$ can be transformed into a purely computational theory in Deduction modulo theory. It also enjoys the disjunction and the witness property, while other formulations only enjoy restricted forms of these properties.

We begin the chapter with the formulation HA^{κ} that uses a sort κ for classes but no predicate symbol N for natural numbers. This formulation of arithmetic will be useful in Chapter 11. We also briefly present the theory HA that uses neither a sort κ for classes nor a predicate N for natural numbers.

We prove that all these formulations are equivalent. The main equivalence notion used is that of *conservative extension*, detailed in Definition 4.3.

4.1 HA^{κ}

Besides the symbols 0 and S, HA^{κ} contains other function symbols: *Pred* for the predecessor function, $+$ for the addition and $\times$ for the multiplication. Its language contains also a unary predicate symbol *Null* and a binary predicate symbol $=$. Some axioms of the theory define the the predecessor function, the addition, the multiplication and the predicate *Null*. The *induction axiom* expresses that there are no other natural numbers than those constructed from 0 and S, that is that every class of numbers containing 0

and closed by successor contains all the natural numbers. To be able to express this axiom we introduce a sort for classes of numbers and axioms expressing the existence of some classes. These classes are also used to express the properties of equality.

Definition 4.1 (HA$^\kappa$) *The language of HA$^\kappa$ contains two sorts ι and κ, function symbols* 0 *of sort ι,* S *and* Pred *of arity $\langle \iota, \iota \rangle$, $+$ and $\times$ of arity $\langle \iota, \iota, \iota \rangle$, predicate symbols* Null *of arity $\langle \iota \rangle$, $=$ of arity $\langle \iota, \iota \rangle$, and ϵ of arity $\langle \iota, \kappa \rangle$. The axioms are*
(the comprehension scheme)
for each proposition A not containing the symbol ϵ and whose free variables are among $x_1, ..., x_n, y$ the axiom

$$\forall x_1 ... \forall x_n \exists c \forall y \; (y \; \epsilon \; c \Leftrightarrow A)$$

(the axiom of equality)

$$\forall x \forall y \; (x = y \Leftrightarrow \forall c \; (x \; \epsilon \; c \Rightarrow y \; \epsilon \; c))$$

(the induction axiom)

$$\forall c \; (0 \; \epsilon \; c \Rightarrow \forall x \; (x \; \epsilon \; c \Rightarrow S(x) \; \epsilon \; c) \Rightarrow \forall y \; y \; \epsilon \; c)$$

(and the axioms defining the predecessor function, the addition, the multiplication and the Null predicate)

$$\text{Pred}(0) = 0$$

$$\forall x \; (\text{Pred}(S(x)) = x)$$

$$\forall y \; (0 + y = y)$$

$$\forall x \; \forall y \; (S(x) + y = S(x + y))$$

$$\forall y \; (0 \times y = 0)$$

$$\forall x \; \forall y \; (S(x) \times y = (x \times y) + y)$$

$$\text{Null}(0)$$

$$\forall x \; \neg \text{Null}(S(x))$$

Exercise 4.1 *Prove the propositions*

$$\forall x \forall y \; (S(x) = S(y) \Rightarrow x = y)$$

$$\forall x \; \neg (0 = S(x))$$

in HA$^\kappa$.

Exercise 4.2 *How are the induction axiom and the comprehension scheme used in the proof of the proposition*

$$\forall x \; (x + 0 = x)$$

in HA$^\kappa$?

Definition 4.2 (Extension) *Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Let $\mathcal{T}$ be an axiomatic theory expressed in $\mathcal{L}$ and $\mathcal{T}'$ one expressed in $\mathcal{L}'$. The theory $\mathcal{T}'$ is said to be an* extension *of $\mathcal{T}$ if all the propositions provable in $\mathcal{T}$ are provable in $\mathcal{T}'$.*

Definition 4.3 (Conservative extension) *Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Let $\mathcal{T}$ be an axiomatic theory expressed in $\mathcal{L}$ and $\mathcal{T}'$ one expressed in $\mathcal{L}'$, that is an extension of $\mathcal{T}$. The theory $\mathcal{T}'$ is said to be a* conservative *extension of $\mathcal{T}$ if all the propositions of $\mathcal{L}$ that are provable in $\mathcal{T}'$ are also provable in $\mathcal{T}$.*

Definition 4.4 (Extension of a model) *Let $\mathcal{L}$ and $\mathcal{L}'$ be two languages such that $\mathcal{L} \subseteq \mathcal{L}'$. Let $\mathcal{M}$ be a model of $\mathcal{L}$ and $\mathcal{M}'$ one of $\mathcal{L}'$. The model $\mathcal{M}'$ is an* extension *of $\mathcal{M}$ if for all sorts s of $\mathcal{L}$ we have $\mathcal{M}_s = \mathcal{M}'_s$, $\mathcal{B} = \mathcal{B}'$, for all function symbols f of $\mathcal{L}$, $\hat{f}^{\mathcal{M}} = \hat{f}^{\mathcal{M}'}$, and for all predicate symbols P of $\mathcal{L}$, $\hat{P}^{\mathcal{M}} = \hat{P}^{\mathcal{M}'}$.*

Proposition 4.1 *Let $\mathcal{L}$ be a language and $\mathcal{T}$ be an axiomatic theory expressed in this language. Let $\mathcal{L}'$ be a language such that $\mathcal{L} \subseteq \mathcal{L}'$ and $\mathcal{T}'$ be an axiomatic theory in $\mathcal{L}'$ such that $\mathcal{T} \subseteq \mathcal{T}'$. If for all models $\mathcal{M}$ of $\mathcal{T}$, there exists an extension $\mathcal{M}'$ of $\mathcal{M}$ that is a model of $\mathcal{T}'$, then $\mathcal{T}'$ is a conservative extension of $\mathcal{T}$.*

Proof. Let A be a proposition in $\mathcal{L}$ that is provable in $\mathcal{T}'$ and $\mathcal{M}$ be any model of $\mathcal{T}$. There exists a model $\mathcal{M}'$ of $\mathcal{T}'$ that is an extension of $\mathcal{M}$. As the model $\mathcal{M}'$ is a model of $\mathcal{T}'$, the proposition A is valid in $\mathcal{M}'$. As $\mathcal{M}'$ is an extension of $\mathcal{M}$, the interpretation of A in $\mathcal{M}$ is the same. Thus, A is valid in $\mathcal{M}$. As it is valid in all models of $\mathcal{T}$, A is provable in $\mathcal{T}$.

Exercise 4.3 (The theory HA) *Consider the language containing a single sort ι, the symbols 0, S,* Pred*, $+$, $\times$,* Null*, and $=$ with the same arity as above and the theory containing*
(the axioms of equality)

$$\forall x\ (x = x)$$

$$\forall x \forall y \forall x' \forall y'\ (x = y \Rightarrow x' = y' \Rightarrow x = x' \Rightarrow y = y')$$

$$\forall x \forall y\ (x = y \Rightarrow S(x) = S(y))$$

$$\forall x \forall y\ (x = y \Rightarrow \text{Pred}(x) = \text{Pred}(y))$$

$$\forall x \forall y \forall x' \forall y'\ (x = y \Rightarrow x' = y' \Rightarrow x + x' = y + y')$$

$$\forall x \forall y \forall x' \forall y'\ (x = y \Rightarrow x' = y' \Rightarrow x \times x' = y \times y')$$

$$\forall x \forall y\ (x = y \Rightarrow \text{Null}(x) \Rightarrow \text{Null}(y))$$

(the induction scheme)
for each proposition A whose free variables are among $x_1, ..., x_n, y$, the proposition

$$\forall x_1 ... \forall x_n\ ((0/y)A \Rightarrow \forall p\ ((p/y)A \Rightarrow (S(p)/y)A) \Rightarrow \forall q\ (q/y)A)$$

(and the axioms defining the predecessor function, the addition, the multiplication and the Null predicate)

$$\text{Pred}(0) = 0$$

$$\forall x\ (\text{Pred}(S(x)) = x)$$

$$\forall y\ (0 + y = y)$$

$$\forall x\ \forall y\ (S(x) + y = S(x + y))$$

$$\forall y\ (0 \times y = 0)$$

$$\forall x\ \forall y\ (S(x) \times y = (x \times y) + y)$$

$$\text{Null}(0)$$

$$\forall x\ \neg\text{Null}(S(x))$$

1. *Show that all the axioms of this theory are provable in HA$^\kappa$.*

2. *Show that HA$^\kappa$ is an extension of this theory.*

3. *Show that for each proposition A whose free variables are among $x_1, ..., x_n, y$, the proposition*

$$\forall x_1 ... \forall x_n\ (z_1 = z_2 \Rightarrow (z_1/y)A \Rightarrow (z_2/y)A)$$

   is provable in this theory.

4. *Show that HA$^\kappa$ is a conservative extension of this theory. (Hint: for $\mathcal{M}_\kappa$ consider the set of definable functions from $\mathcal{M}_\iota$ to $\mathcal{B}$, that is those of the form $a \mapsto [\![A]\!]_{\phi, x=a}$ for some proposition A not using the symbol ϵ and valuation ϕ.)*

Remark. In HA$^\kappa$, and in other formulations of arithmetic below, the comprehension scheme

$$\forall x_1 ... \forall x_n \exists c \forall y\ (y\ \epsilon\ c \Leftrightarrow A)$$

is stated for propositions A that do not contain the symbol ϵ, that is for propositions that contain no terms of sort κ and hence no non-dummy quantifiers on the sort κ.

This restriction can be dropped and we may include instances of the scheme for propositions containing the symbol ϵ, variables of sort κ, and quantifiers on the sort κ. We get this way another formulation of arithmetic, called *impredicative arithmetic* or sometimes *second-order arithmetic*, in contrast HA$^\kappa$ is said to be *predicative* or *first-order*.

Impredicative and predicative arithmetic are different theories. In particular, impredicative arithmetic is not a conservative extension of predicative arithmetic.

4.2 Peano's predicate symbol

The induction axiom of HA^κ expresses that every class that contains zero and that is closed by successor contains everything. Thus all objects of sort ι are natural numbers. An alternative is to allow objects that are not natural numbers, and to introduce a predicate symbol N for the natural numbers. If we introduce such a symbol, we must extend the comprehension scheme to propositions A containing the symbol N, we must add axioms expressing that 0 is a natural number and that if x is a natural number, so is $S(x)$

$$N(0)$$

$$\forall x\ (N(x) \Rightarrow N(S(x)))$$

and the induction axiom must express that every class containing 0 and closed by successor contains, not all the objects, but all the objects that are natural numbers

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ (N(y) \Rightarrow y \epsilon c))$$

In this axiom, we can also relax the condition that c must be closed by the successor function: instead of requiring c to contain $S(x)$ whenever it contains x, we can require it to contain $S(x)$ whenever it contains x and x is a natural number

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ (N(y) \Rightarrow y \epsilon c))$$

Exercise 4.4 *Consider the axiom*

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ (N(y) \Rightarrow y \epsilon c))$$

and prove

$$(0/z)A \Rightarrow \forall x\ (N(x) \Rightarrow (x/z)A \Rightarrow (S(x)/z)A) \Rightarrow \forall y\ (N(y) \Rightarrow (y/z)A)$$

Consider the axiom

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\ (N(y) \Rightarrow y \epsilon c))$$

and prove

$$(0/z)A \Rightarrow \forall x\ (N(x) \Rightarrow (x/z)A \Rightarrow (S(x)/z)A) \Rightarrow \forall y\ (N(y) \Rightarrow (y/z)A)$$

Both formulations of the induction axiom are possible, but we shall use the first, that will lead to a slightly more difficult super-consistency proof in this chapter, but that will happen to be more useful when we prove the termination of Gödel's system T in Chapter 11.

The induction axiom is equivalent to

$$\forall y\ (N(y) \Rightarrow \forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c))$$

and we may remark that, thanks to the axioms $N(0)$ and $\forall x\ (N(x) \Rightarrow N(S(x)))$, the converse of this statement is provable. Indeed, if we assume

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c))$$

then, using the comprehension scheme, we prove the existence of a class $\mathbb{N}$ such that $\forall x\ (x \epsilon \mathbb{N} \Leftrightarrow N(x))$ and instantiating c with this class we get a proposition equivalent to

$$N(0) \Rightarrow \forall x\ (N(x) \Rightarrow N(x) \Rightarrow N(S(x))) \Rightarrow N(y)$$

and then, using the axioms $N(0)$ and $\forall x\ (N(x) \Rightarrow N(S(x)))$, we get $N(y)$. Thus, the induction axiom can be also stated

$$\forall y\ (N(y) \Leftrightarrow \forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c))$$

If we take this formulation of the axiom, then the axioms $N(0)$ and $\forall x\ (N(x) \Rightarrow N(S(x)))$ are not needed anymore. For instance, the proposition $N(0)$ is equivalent to

$$\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow 0 \epsilon c)$$

that is obviously provable and the same holds for the axiom

$$\forall x\ (N(x) \Rightarrow N(S(x)))$$

This leads to the following definition of the theory $\mathrm{HA}^{\kappa N}$.

Definition 4.5 ($\mathrm{HA}^{\kappa N}$) *The language of $\mathrm{HA}^{\kappa N}$ contains two sorts ι and κ, function symbols 0 of sort ι, S and* Pred *of arity $\langle \iota, \iota \rangle$, $+$ and $\times$ of arity $\langle \iota, \iota, \iota \rangle$ and predicate symbols* Null *of arity $\langle \iota \rangle$, $=$ of arity $\langle \iota, \iota \rangle$, N of arity $\langle \iota \rangle$, and ϵ of arity $\langle \iota, \kappa \rangle$. The axioms are*
(the comprehension scheme)
for each proposition A not containing the symbol ϵ and whose free variables are among $x_1, ..., x_n, y$ the axiom

$$\forall x_1 ... \forall x_n \exists c \forall y\ (y \epsilon c \Leftrightarrow A)$$

(the axiom of equality)

$$\forall x \forall y\ (x = y \Leftrightarrow \forall c\ (x \epsilon c \Rightarrow y \epsilon c))$$

(the induction axiom)

$$\forall y\ (N(y) \Leftrightarrow \forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c))$$

(and the axioms defining the predecessor function, the addition, the multiplication and the Null predicate)

$$\mathrm{Pred}(0) = 0$$

$$\forall x\ (\mathrm{Pred}(S(x)) = x)$$

$$\forall y\ (0 + y = y)$$

$$\forall x\ \forall y\ (S(x) + y = S(x + y))$$

$$\forall y\ (0 \times y = 0)$$

$$\forall x\ \forall y\ (S(x) \times y = (x \times y) + y)$$

$$\mathrm{Null}(0)$$

$$\forall x\ \neg \mathrm{Null}(S(x))$$

We get this way a slightly more complex theory but, as we shall see, this theory enjoys the disjunction property and the witness property, while HA^{κ} only enjoys a restricted form of these properties. Indeed, in HA^{κ}, we can prove by induction the proposition

$$\forall x\ (x = 0 \vee \exists y\ (x = S(y)))$$

and from this proposition, we can deduce the proposition with a free variable x

$$x = 0 \vee \exists y\ (x = S(y))$$

But, of course, neither the proposition $x = 0$ nor the proposition $\exists y\ (x = S(y))$ is provable. Thus, the disjunction property and the witness property have to be restricted to closed propositions. In $\mathrm{HA}^{\kappa N}$, the proposition

$$x = 0 \vee \exists y\ (x = S(y))$$

is not provable. The proposition

$$N(x) \Rightarrow (x = 0 \vee \exists y\ (x = S(y)))$$

is, but it is not a disjunction.

$\mathrm{HA}^{\kappa N}$ is not an extension of HA^{κ} because the proposition

$$\forall x\ (x = 0 \vee \exists y\ (x = S(y)))$$

is provable in the latter but not in the former. But, a closed proposition A is provable in HA^{κ} if and only if its translation $|A|$ is provable in $\mathrm{HA}^{\kappa N}$, where the translation $|\ |$ is defined as follows.

Definition 4.6 (Translation)

- $|P| = P$, *if* P *is atomic,*
- $|\top| = \top$, $|\bot| = \bot$,
- $|A \wedge B| = |A| \wedge |B|$, $|A \vee B| = |A| \vee |B|$, $|A \Rightarrow B| = |A| \Rightarrow |B|$,
- $|\forall x\ A| = \forall x\ (N(x) \Rightarrow |A|)$, $|\exists x\ A| = \exists x\ (N(x) \wedge |A|)$,
- $|\forall c\ A| = \forall c\ |A|$, $|\exists c\ A| = \exists c\ |A|$.

Proposition 4.2 *If* A *is an axiom of* HA^{κ} *symbol then* $|A|$ *is provable in* $\mathrm{HA}^{\kappa N}$.

Proof. We check the axioms one by one.

Proposition 4.3 *The propositions*

1. $N(0)$
2. $\forall x\ (N(x) \Rightarrow N(S(x)))$
3. $\forall x\ (N(x) \Rightarrow N(\mathrm{Pred}(x)))$

4. $\forall x\ \forall y\ (N(x) \Rightarrow N(y) \Rightarrow N(x+y))$

5. $\forall x\ \forall y\ (N(x) \Rightarrow N(y) \Rightarrow N(x \times y))$

are provable in HA$^{\kappa N}$.

Proof.

1. The proposition $N(0)$ is equivalent to

$$\forall c\ (0\ \epsilon\ c \Rightarrow \forall x\ (N(x) \Rightarrow x\ \epsilon\ c \Rightarrow S(x)\ \epsilon\ c) \Rightarrow 0\ \epsilon\ c)$$

that is obviously provable.

2. Same a case *1.*

3. An instance of the induction scheme is

$$N(\mathit{Pred}(0))) \Rightarrow \forall x\ (N(x) \Rightarrow N(\mathit{Pred}(x)) \Rightarrow (N(\mathit{Pred}(S(x))))) \Rightarrow \forall x\ (N(x) \Rightarrow N(\mathit{Pred}(x)))$$

and the propositions $N(\mathit{Pred}(0))$, $\forall x\ (N(x) \Rightarrow N(\mathit{Pred}(x)) \Rightarrow N(\mathit{Pred}(S(x))))$, and $N(x)$ can be proved from the hypothesis $N(x)$. Thus the proposition $N(\mathit{Pred}(x))$ can be proved from the hypothesis $N(x)$.

4. An instance of the induction scheme is

$$N(0{+}y) \Rightarrow \forall x\ (N(x) \Rightarrow N(x{+}y) \Rightarrow (N(S(x){+}y))) \Rightarrow \forall x\ (N(x) \Rightarrow N(x{+}y))$$

and the propositions $N(0+y)$, $\forall x\ (N(x) \Rightarrow N(x+y) \Rightarrow (N(S(x)+y)))$, and $N(x)$ can be proved from the hypotheses $N(x)$ and $N(y)$. Thus the proposition $N(x+y)$ can be proved from the hypotheses $N(x)$ and $N(y)$.

5. Same as case *4.*

Proposition 4.4 *Let t be a term whose variables among $x_1, ..., x_n$. Then the sequent $N(x_1), ..., N(x_n) \vdash N(t)$ is provable in HA$^{\kappa N}$.*

Proof. By induction on term structure, using Proposition 4.3.

Proposition 4.5 *If the sequent $\Gamma, N(x) \vdash A$ is provable in HA$^{\kappa N}$ and x is free neither in Γ nor in A, then the sequent $\Gamma \vdash A$ is provable in this same theory.*

Proof. As the proposition $N(0)$ is provable in HA$^{\kappa N}$, so is the proposition $\exists x\ N(x)$. Thus from a proof of $\Gamma, N(x) \vdash A$ we can build a proof of $\Gamma \vdash A$ with the elimination rule of the existential quantifier.

Proposition 4.6 *Let A be a closed proposition in the language of HA$^{\kappa}$, if A is provable in HA$^{\kappa}$, then $|A|$ is provable in HA$^{\kappa N}$.*

Proof. We prove, more generally, that if the sequent $\Gamma \vdash A$ is provable in HA$^\kappa$, then there exists a sequence of variables $x_1, ..., x_n$ such that the sequent $|\Gamma|, N(x_1), ..., N(x_n) \vdash |A|$ is provable in HA$^{\kappa N}$. The result follows using Proposition 4.5. We proceed by induction on the structure of the proof of the sequent $\Gamma \vdash A$ using Proposition 4.2 for axioms and Proposition 4.4 for quantifier rules.

For instance, if the last rule of the proof is the elimination of the universal quantifier

$$\dfrac{\dfrac{\pi}{\Gamma \vdash \forall x\ C}}{\Gamma \vdash (t/x)C}$$

then, by induction hypothesis, the sequent $|\Gamma|, N(x_1), ..., N(x_n) \vdash \forall x\ (N(x) \Rightarrow |C|)$ is provable. Let $y_1, ..., y_p$ be the variables of t. By weakening, the sequent $N(x_1), ..., N(x_n), N(y_1), ..., N(y_p), |\Gamma| \vdash \forall x\ (N(x) \Rightarrow |C|)$ is provable. By Proposition 4.4, the sequent $N(x_1), ..., N(x_n), N(y_1), ..., N(y_p), |\Gamma| \vdash N(t)$ is provable. Using an elimination of the universal quantifier and an elimination of the implication, we build a proof of $N(x_1), ..., N(x_n), N(y_1), ..., N(y_p), |\Gamma| \vdash |(t/x)C|$.

Proposition 4.7 *Let A be a closed proposition in the language of HA$^\kappa$. If $|A|$ is provable in HA$^{\kappa N}$ then A is provable in HA$^\kappa$.*

Proof. We first prove that any model $\mathcal{M}$ of HA$^\kappa$ can be extended in a model $\mathcal{M}'$ of HA$^{\kappa N}$ and, moreover, that if A is a proposition in the language of HA$^\kappa$, then A is valid in $\mathcal{M}'$ if and only if $|A|$ is.

Let $\mathcal{M}$ be any model of HA$^\kappa$. Let $\mathcal{M}'$ be the model whose domains $\mathcal{M}'_\iota$ and $\mathcal{M}'_\kappa$ are the same as in $\mathcal{M}$, whose pre-Heyting algebra is the same as in $\mathcal{M}$, and where the interpretation of the symbols of HA$^\kappa$ is the same as in $\mathcal{M}$ and where $\hat{N}$ is the function mapping every element of $\mathcal{M}_\iota$ to $\tilde{\top}$. We check, axiom by axiom, that the model $\mathcal{M}'$ is a model of HA$^{\kappa N}$.

If A is a proposition in the language of HA$^\kappa$, we prove by induction on the structure of A that $[\![A]\!]^{\mathcal{M}'}_\phi \leq [\![|A|]\!]^{\mathcal{M}'}_\phi$ and $[\![|A|]\!]^{\mathcal{M}'}_\phi \leq [\![A]\!]^{\mathcal{M}'}_\phi$. Thus $|A|$ is valid in $\mathcal{M}'$ if and only if A is.

The end of the proof follows that of Proposition 4.1. Let A be a proposition in HA$^\kappa$, such that $|A|$ is provable in HA$^{\kappa N}$ and $\mathcal{M}$ be any model of HA$^\kappa$. Let $\mathcal{M}'$ be the extension of $\mathcal{M}$ built above, $\mathcal{M}'$ is a model of HA$^{\kappa N}$. As the model $\mathcal{M}'$ is a model of HA$^{\kappa N}$ and $|A|$ is provable in this theory, $|A|$ is valid in $\mathcal{M}'$, hence A is valid in $\mathcal{M}'$ and as $\mathcal{M}'$ is an extension of $\mathcal{M}$, the interpretation of A in $\mathcal{M}$ is the same. Thus, A is valid in $\mathcal{M}$.

As it is valid in all models of $\mathcal{T}$, the proposition A is provable in $\mathcal{T}$.

4.3 Arithmetic as a purely computational theory

We now want to replace all the axioms of HA$^{\kappa N}$ by reduction rules defining a congruence. The axioms defining the predecessor function, the addition, the multiplication and the Null predicate are easy to replace by reduction rules

$$\mathit{Pred}(0) \longrightarrow 0$$

$$\mathit{Pred}(S(x)) \longrightarrow x$$
$$0 + y \longrightarrow y$$
$$S(x) + y \longrightarrow S(x + y)$$
$$0 \times y \longrightarrow 0$$
$$S(x) \times y \longrightarrow (x \times y) + y$$
$$\mathit{Null}(0) \longrightarrow \top$$
$$\mathit{Null}(S(x)) \longrightarrow \bot$$

The axiom of equality can be seen as a definition of equality: x and y are equal if they belong to the same classes. The induction axiom can be seen as a sort of definition of the predicate N: natural numbers are what belong to all classes containing zero and closed by the successor function, although formally this is not a definition because the symbol N occurs on the right-hand side. We can easily orient them as reduction rules

$$x = y \longrightarrow \forall c\ (x \in c \Rightarrow y \in c)$$

$$N(y) \longrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (N(x) \Rightarrow x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$

Finally, to replace the comprehension scheme by reduction rules, we shall introduce for each proposition A not containing the symbol ϵ and whose free variables are among $x_1, ..., x_n, y$, a function symbol $f_{x_1,...,x_n,y,A}$. The term $f_{x_1,...,x_n,y,A}(x_1, .., x_n)$ is a notation for the class defined in comprehension by the proposition A and we take the rule

$$y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$

Note that the free variables are the same in the left-hand side and in the right-hand side of the rule.

Definition 4.7 (HA$^{\longrightarrow}$) *The language of HA$^{\longrightarrow}$ contains two sorts ι and κ, function symbols 0 of sort ι, S and* Pred *of arity $\langle \iota, \iota \rangle$, $+$ and $\times$ of arity $\langle \iota, \iota, \iota \rangle$, predicate symbols $=$ of arity $\langle \iota, \iota \rangle$,* Null *and N of arity $\langle \iota \rangle$, and ϵ of arity $\langle \iota, \kappa \rangle$, and for each proposition A in the language 0, S,* Pred*, $+$, $\times$, $=$,* Null *and N and whose free variables are among $x_1, ..., x_n, y$, a function symbol $f_{x_1,...,x_n,A}$ of arity $\langle \iota, \ldots, \iota, \kappa \rangle$. The reduction rules are*

$$y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$
$$x = y \longrightarrow \forall c\ (x \in c \Rightarrow y \in c)$$
$$N(y) \longrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (N(x) \Rightarrow x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$
$$\text{Pred}(0) \longrightarrow 0$$
$$\text{Pred}(S(x)) \longrightarrow x$$
$$0 + y \longrightarrow y$$
$$S(x) + y \longrightarrow S(x + y)$$
$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow x \times y + y$$
$$\text{Null}(0) \longrightarrow \top$$
$$\text{Null}(S(x)) \longrightarrow \bot$$

Proposition 4.8 *The theory HA$^{\longrightarrow}$ is a conservative extension of HA$^{\kappa N}$.*

Proof. We first prove that the extension of HA$^{\kappa N}$, obtained by adding the symbols $f_{x_1,...,x_n,y,A}$ and replacing the comprehension scheme by the scheme

$$\forall x_1 ... \forall x_n \forall y \; (y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \Leftrightarrow A)$$

is a conservative extension of HA$^{\kappa N}$. The comprehension scheme is obviously provable from this scheme. Thus this theory is an extension of HA$^{\kappa N}$. To prove that this extension is conservative, we use Proposition 4.1 and prove that any model of HA$^{\kappa N}$ can be extended into a model of this theory.

Let $\mathcal{M}$ be a model of HA$^{\kappa N}$. As this model is a model of the comprehension scheme, we know that, for each proposition A whose free variables are among $x_1, ..., x_n, y$ and sequence $a_1, .., a_n$ of element of $\mathcal{M}_\iota$, there exists an element b of $\mathcal{M}_\kappa$ such that $[\![\forall y \; (y \in c \Leftrightarrow A)]\!]_{x_1=a_1,...,x_n=a_n,c=b} \geq \tilde{\top}$. We extend the model $\mathcal{M}$, by defining $\hat{f}_{x_1,...,x_n,y,A}$ as the function mapping $a_1, ..., a_n$ to b. Then, we have $[\![\forall y \; (y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n)) \Leftrightarrow A)]\!]_{x_1=a_1,...,x_n=a_n} = [\![\forall y \, (y \in c \Leftrightarrow A)]\!]_{x_1=a_1,...,x_n=a_n,c=b} \geq \tilde{\top}$. Thus, this model is a model of the axioms

$$\forall x_1 ... \forall x_n \forall y \; (y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \Leftrightarrow A)$$

Finally, we conclude by remarking that the proposition provable in this theory and in HA$^{\longrightarrow}$ are the same.

4.4 Models of arithmetic

Proposition 4.9 *The theory HA$^{\longrightarrow}$ has a model valued in the algebra $\{0, 1\}$. Hence it is consistent.*

Proof. Let $\mathcal{M}_\iota = \mathbb{N}$ and $\mathcal{M}_\kappa = \mathbb{N} \to \{0, 1\}$. We define $\hat{0}$, $\hat{S}$, $\hat{Pred}$, $\hat{+}$, $\hat{\times}$ in the obvious way, $\hat{Null}$ be the function mapping 0 to 1 and the other numbers to 0, $\hat{\epsilon}$ be function mapping the number n and the function g of $\mathbb{N} \to \{0, 1\}$ to $g(n)$, $\hat{=}$ be the function mapping n and p to 1 if $n = p$ and to 0 otherwise, $\hat{N}$ be the constant function equal to 1, and finally $\hat{f}_{x_1,...,x_n,y,A}$ be the function mapping $a_1, ..., a_n$ to the function mapping b to $[\![A]\!]_{x_1=a_1,...,x_n=a_n,y=b}$. It is routine to check that all rules of arithmetic are valid in this model.

We now want to prove that arithmetic is super-consistent. Assume we are given a full, ordered and complete pre-Heyting algebra $\mathcal{B}$, we can imitate the proof of Proposition 4.9 and build a model whose pre-Heyting algebra is $\mathcal{B}$, by taking $\mathcal{M}_\iota = \mathbb{N}$, $\mathcal{M}_\kappa = \mathbb{N} \to \mathcal{B}$, $\hat{0}$, $\hat{S}$, $\hat{Pred}$, $\hat{+}$, $\hat{\times}$, as above, $\hat{Null}$ be the function mapping 0 to $\tilde{\top}$ and the other numbers to $\tilde{\bot}$, and $\hat{\epsilon}$ to be function mapping the number n and the function

g of $\mathbb{N} \to \mathcal{B}$ to $g(n)$. All the rules defining the predecessor function, the addition, the multiplication and the Null predicate will be valid in this model.

The only symbols that remain to be interpreted are $=$, N, and $f_{x_1,...,x_n,y,A}$ and their interpretation has to be chosen in order to validate the three remaining rules

$$y \epsilon f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$

$$x = y \longrightarrow \forall c\, (x \epsilon c \Rightarrow y \epsilon c)$$

$$N(y) \longrightarrow \forall c\, (0 \epsilon c \Rightarrow \forall x\, (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c)$$

As these rules can be seen as definitions, we may be tempted to interpret their left-hand side as their right-hand side is already interpreted. This is, for instance, possible for the rule

$$x = y \longrightarrow \forall c\, (x \epsilon c \Rightarrow y \epsilon c)$$

The interpretation of the proposition $\forall c\, (x \epsilon c \Rightarrow y \epsilon c)$ is already defined as the only symbol occurring in it is ϵ. Thus, we may define $\hat{=}$ as the function mapping n and p to $[\![\forall c\, (x \epsilon c \Rightarrow y \epsilon c)]\!]_{x=n,y=p}$, that is $\tilde{\forall}\, \{f(n) \tilde{\Rightarrow} f(p) \mid f \in \mathbb{N} \to \mathcal{B}\}$, and this way the rule of equality will be valid.

But we cannot do this for the induction rule, because the symbol N occurs also in the right-hand side. Here we see the drawback of having taken the induction axiom

$$\forall c\, (0 \epsilon c \Rightarrow \forall x\, (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\, (N(y) \Rightarrow y \epsilon c))$$

and not

$$\forall c\, (0 \epsilon c \Rightarrow \forall x\, (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow \forall y\, (N(y) \Rightarrow y \epsilon c))$$

Yet, in the definition of super-consistency, we have been wise enough to restrict to ordered and complete pre-Heyting algebras. Using the completeness of the pre-Heyting algebra $\mathcal{B}$, we will be able to build an interpretation of the symbol N, as a fixed-point.

Proposition 4.10 *HA$^{\longrightarrow}$ is super-consistent.*

Proof. Let $\mathcal{B}$ be a full, ordered and complete pre-Heyting algebra. We let $\mathcal{M}_\iota = \mathbb{N}$, $\mathcal{M}_\kappa = \mathbb{N} \to \mathcal{B}$. The interpretations of the symbols 0, S, $+$, $\times$, and *Pred* are defined in the obvious way. We let $\hat{Null}$ be the function mapping 0 to $\tilde{\top}$ and the other numbers to $\tilde{\bot}$. The interpretation of ϵ is the function mapping n and g to $g(n)$. Then, we can define the interpretation of $\forall c\, (y \epsilon c \Rightarrow z \epsilon c)$ and we define $\hat{=}$ as the function mapping n and p to $[\![\forall c\, (x \epsilon c \Rightarrow y \epsilon c)]\!]_{x=n,y=p}$, that is $\tilde{\forall}\, \{f(n) \tilde{\Rightarrow} f(p) \mid f \in \mathbb{N} \to \mathcal{B}\}$.

To define the interpretation of N, for each function f of $\mathbb{N} \to \mathcal{B}$ we can define an interpretation $\mathcal{M}_f$ of the language of the proposition

$$\forall c\, (0 \in c \Rightarrow \forall x\, (N(x) \Rightarrow x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$

where the symbol N is interpreted by the function f. We define the function Φ from $\mathbb{N} \to \mathcal{B}$ to itself mapping f to the function mapping the natural number n to

$$[\![\forall c\, (0 \in c \Rightarrow \forall x\, (N(x) \Rightarrow x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)]\!]^{\mathcal{M}_f}_{n/y}$$

The order on $\mathbb{N} \to \mathcal{B}$ defined by $f \sqsubseteq g$ if for all n, $f(n) \sqsubseteq g(n)$ is a complete order and the function Φ is monotone as the occurrence of N is positive in

$$\forall c\,(0 \in c \Rightarrow \forall x\,(N(x) \Rightarrow x \in c \Rightarrow S(y) \in c \Rightarrow y \in c)$$

Hence it has a fixed-point g. We interpret the symbol N by the function g.

Finally, the interpretation of the symbols of the form $f_{x,y_1,\ldots,y_n,A}$ is defined in the obvious way.

Exercise 4.5 *Leivant, Krivine, and Parigot have proposed to extend arithmetic with rules similar to those of addition and multiplication for other functions. For instance the rules*

$$\chi(0) \longrightarrow 0$$
$$\chi(S(n)) \longrightarrow S(0)$$

that define the function χ that takes the value 0 at 0 and the value 1 at other natural numbers.

Consider an extension of $HA^{\longrightarrow}$ with a symbol χ and the two rules above. Prove that this extension of $HA^{\longrightarrow}$ is super-consistent. (Hint: just add a few characters to the proof of Theorem 4.10.)

Chapter 5

Naive set theory

In arithmetic, we can express and prove many properties of the natural numbers. But there are more objects in mathematics than the natural numbers: integers, rational numbers, real numbers, points, lines, vectors, etc. Designing a theory where all mathematics could be expressed was the central problem in the logic of the end of the 19^{th} century. These investigations lead to the definition of two theories *Set theory* and *Simple type theory*, also called *Higher-order logic*.

We know that, to express mathematics, all we need is the natural numbers plus sets and functions, as all the other objects: integers, rational numbers, real numbers, points, lines, vectors, etc. can be built from these. For instance, real numbers can be built as sets of Cauchy sequences, that are functions mapping natural numbers to rational numbers. Thus, to extend arithmetic into a theory where all mathematics can be expressed, we have to focus on sets and functions. In fact, sets and functions are not both needed as functions can be defined as functional relations, that is set of ordered pairs and, in a symmetric way, sets can be defined as their characteristic function. In arithmetic, terms of sort ι express natural numbers. The terms of sort κ introduced to facilitate the formulation of the induction axiom express classes, that is sets, of natural numbers. But, there is no way to express sets of sets of natural numbers, or functions mapping natural numbers to natural numbers.

5.1 Application and membership

The language of arithmetic contains function symbols, such as S or *Pred*, that express functions mapping natural numbers to natural numbers. These symbols are used to build terms, but they are not terms themselves. For instance, the symbol S is used to build the term $S(0)$, but it is not itself a term. In a language where functions are objects, such a symbol should be a constant, that is a function symbol of arity 0. When this symbol is a constant, the term $S(0)$ can no more be built, because, unlike a unary function symbol, a constant cannot be applied to a term. A new symbol α must be introduced for function application and the term formerly written $S(0)$ must be written $\alpha(S, 0)$. This symbol α is the counterpart for functions of the symbol ϵ introduced for

classes: when a predicate symbol *even* is replaced by a constant we must write $0 \; \epsilon \; \mathit{even}$ what was previously written $\mathit{even}(0)$.

For functions of several arguments, such as $+$ or $\times$, similar function symbols α_2, α_3, ... must be introduced, in order to write $\alpha_2(+, x, 0)$ for the result of applying the function $+$ to the terms x and 0. But, such symbols are not needed because a function f of n arguments can always be seen as a function of one argument mapping x to the function mapping $x_2, ..., x_n$ to $f(x, x_2, ..., x_n)$. Hence, instead of writing $\alpha_n(f, x_1, ..., x_n)$, we can write $\alpha(...\alpha(f, x_1)..., x_n)$. We shall often write $(f \; x)$ for the term $\alpha(f, x)$ and $(f \; x_1 \; ... \; x_n)$ for the term $(...(f \; x_1)...x_n)$.

In Chapter 4 the symbol ϵ was used to relate a natural number and a class of natural numbers. This symbol must be generalized, in order to be able to relate two arbitrary objects, such as two sets, to express that one is an element of the other. In this case, it is more usual to write this symbol $\in$. To handle, not only sets, but also relations, symbols $\in_2$, $\in_3$, ... should be introduced, in order to write, for instance, the proposition $\in_2 (\leq, x, y)$.

A symbol $\in_0$, also written ε, must also be introduced in order to be able to build the proposition $\varepsilon(t)$ from a term t expressing a relation with no arguments. There is little difference between the term t expressing a relation with no arguments and the proposition $\varepsilon(t)$, the only one being that the term t is a term, hence expressing an object, while the proposition $\varepsilon(t)$ is a proposition, hence expressing a fact. The term t is called the *propositional content* of the proposition $\varepsilon(t)$.

The notions of function and set are redundant: a function can always be defined as a set of pairs, and, conversely, a set or a relation can be defined by its characteristic function. Expressing a set E as its characteristic function, that is as the function mapping its argument x to the propositional content of the proposition expressing that x is an element of E, leads to write $\varepsilon(E \; x)$ the proposition written $x \in E$ when the notion of set is primitive, and similarly $\varepsilon(R \; x \; y)$ the proposition $\in_2 (R, x, y)$. Thus, the symbols $\in$, $\in_2$, ... are not needed and the only needed symbols are α and ε.

5.2 Building functions and sets

We can now turn to the symbols that must be introduced to build functions and sets. Informally, a function is expressed by a term that contains a free variable, for example $3 \times x$. But this notation is ambiguous as it does not distinguish *the result of the function* from the *function* itself. For instance, when we say that $3 \times x$ is a multiple of 3, we speak about the result of the function, when we say that $3 \times x$ is monotone, we speak about the function itself. To avoid this ambiguity, we often write the function, not $3 \times x$, but $x \mapsto 3 \times x$. But, in informal mathematics, we are sometimes reluctant to use this symbol, except in a definition: we are used to write the definition

$$f = (x \mapsto 3 \times x)$$

then, the terms $(f \; 4)$ and $\int_0^1 f$, but we usually do not write terms such as $((x \mapsto 3 \times x) \; 4)$ or $\int_0^1 (x \mapsto 3 \times x)$. Here, this notation will be used more systematically and these four terms will be used. Thus, for each term t, whose free variables are

among $x_1, ..., x_n$, we introduce a constant $x_1, ..., x_n \mapsto t$. This constant is called a *combinator*.

In Chapter 4, we have introduced function symbols $f_{x_1,...,x_n,y,A}$ such that the term $f_{x_1,...,x_n,y,A}(x_1, ..., x_n)$ expresses the class of objects y verifying the property A. We extend this notation in order to define a relation of several arguments by a proposition A containing several variables, but we restrict it in such a way that this proposition does not contain any free variables besides those. Thus, for each proposition A, whose free variables are among $x_1, ..., x_n$, we introduce a constant $\{x_1, ..., x_n \mid A\}$, also called a *combinator*. In these definitions, no combinator can occur in the term t or in the proposition A.

5.3 Axioms and rules

When we apply the function $x \mapsto (x \times x) + 2$, for instance to the term 7, we want to obtain a term that is equal to $(7 \times 7) + 2$. In the same way, when we apply the set $\{x \mid \exists y\ (x = 2 \times y)\}$ to the term 7, we want to obtain a proposition equivalent to $\exists y\ (7 = 2 \times y)$. We thus state the *conversion axioms*:

$$\forall x_1 ... \forall x_n\ (((x_1, ..., x_n \mapsto t)\ x_1\ ...\ x_n) = t)$$

$$\forall x_1 ... \forall x_n\ (\varepsilon(\{x_1, ..., x_n \mid A\}\ x_1\ ...\ x_n) \Leftrightarrow A)$$

In Deduction modulo theory, these axioms are turned into *conversion rules*

$$((x_1, ..., x_n \mapsto t)\ x_1\ ...\ x_n) \longrightarrow t$$

$$\varepsilon(\{x_1, ..., x_n \mid A\}\ x_1\ ...\ x_n) \longrightarrow A$$

In an alternative formulation, we do not introduce symbols for these functions, sets and relations, but we just give axioms expressing their existence: the *comprehension axioms*. For each term t of the language, we take the axiom

$$\exists f\ \forall x_1\ ...\ \forall x_n\ ((f\ x_1\ ...\ x_n) = t)$$

and for each proposition A the axiom

$$\exists E\ \forall x_1\ ...\ \forall x_n\ (\varepsilon(E\ x_1\ ...\ x_n) \Leftrightarrow A)$$

but there seems to be no way to transform these axioms into reduction rules: reduction rules require a notation for the objects, as otherwise there is nothing to reduce.

5.4 Russell's paradox

Several theories close to this one have been proposed in the history of logic, for instance Cantor's set theory (1872), Frege's *Begriffschrift* (1879) or Church's system based on untyped λ-calculus (1932). Unfortunately all these systems are inconsistent as shown by Russell's paradox.

Let $R = \{x \mid \neg\varepsilon(x\ x)\}$ be the set of the sets that are not elements of themselves. By definition the set R is an element of R if and only if it is not an element of itself, which is inconsistent. More precisely, the proposition *the set R is an element of R* is $A = \varepsilon(R\ R) = \varepsilon(\{x \mid \neg\varepsilon(x\ x)\}\ R)$. This proposition reduces to $\neg\varepsilon(R\ R)$, that is to $\neg A$. Thus, we can prove $\neg A$, then A, and $\bot$.

5.5 Type theory and set theory

In naive set theory,

- every predicate is an object,
- every predicate can be applied to every object.

The conjunction of these two principles leads to Russell's paradox. To avoid this paradox, two ways are possible: abandon the first principle or abandon the second. This two ways lead to two different theories: Set theory and Simple type theory.

5.6 Set theory

5.6.1 Term formation

In Zermelo *set theory* (1908), and its extension *Zermelo-Fraenkel* set theory, functions are not primitive objects, but relations. Relations are not primitive either, but sets of ordered pairs. Thus, the only primitive notion is that of set and, besides equality, the only predicate is the membership predicate $\in$.

In this language Russell's paradox is expressed as follows: the set R is expressed as $\{x \mid \neg x \in x\}$ and the proposition *the set R is an element of R* as $A = R \in R = \{x \mid \neg x \in x\} \in \{x \mid \neg x \in x\}$. And this proposition reduces to $\neg A$.

The idea of set theory is to abandon the principle according to which every predicate is an object. When A is a proposition, it is not always possible to build the set $\{x \mid A\}$. In other words, it is not always possible to build a set *in comprehension*, that is by giving a characteristic property of its elements. This is only possible in the following cases.

- If E and F are two sets, then we can form a set whose elements are E and F: *the pair of E and F*.
- If E is a set, then we can form a set whose elements are the elements of the elements of E: *the union of the elements of E*.
- If E is a set, then we can form a set whose elements are the subsets of E: *the powerset of E*.
- If E is a set and A a proposition in the language $=, \in$, then we can form a set whose elements are the elements of E verifying A: *the subset of E of the elements verifying A*.

For this last construction, it is convenient to introduce a sort κ for classes of sets and a comprehension scheme expressing that every proposition in the language $=, \in$ defines a class in comprehension.

Thus, we introduce function symbols $\{,\}, \bigcup, \mathcal{P}$, and $\{|\}$ and the *conversion axioms*

$$\forall E \forall F \forall x \, (x \in \{E, F\} \Leftrightarrow (x = E \vee x = F))$$

$$\forall E \forall x \, (x \in \bigcup(E) \Leftrightarrow \exists y \, (x \in y \wedge y \in E))$$

$$\forall E \forall x \, (x \in \mathcal{P}(E) \Leftrightarrow \forall y \, (y \in x \Rightarrow y \in E))$$

$$\forall E \forall c \forall x (x \in \{E \mid c\} \Leftrightarrow (x \in E \wedge x \, \epsilon \, c))$$

Finally, we introduce function symbols $f_{x_1,...,x_n,y,A}$ to construct classes and an axiom scheme

$$\forall x_1 ... \forall x_n \forall y \, (y \, \epsilon \, f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \Leftrightarrow A)$$

In the fourth axiom, remark the difference of notation: $x \in E$ expresses that the set x is an element of the set E and $x \, \epsilon \, c$ that it is an element of the class c.

This way, there is no way to construct the set of sets that are not element of themselves. The question whether a set is an element of itself is, in contrast, always well-formed. For instance, if E is a set and $c = f_{y,\perp}$ is the empty class, then the empty subset of E, $\{E \mid c\}$, is not an element of itself, as the proposition $\neg(\{E \mid c\} \in \{E \mid c\})$ is equivalent to $\neg(\{E \mid c\} \in E \wedge \{E \mid c\} \, \epsilon \, c)$ and then to $\neg(\{E \mid c\} \in E \wedge \perp)$ that is provable.

These axioms may be transformed into reduction rules

$$x \in \{E, F\} \longrightarrow (x = E \vee x = F)$$

$$x \in \bigcup(E) \longrightarrow \exists y \, (y \in E \wedge x \in y)$$

$$x \in \mathcal{P}(E) \longrightarrow \forall y \, (y \in x \Rightarrow y \in E)$$

$$x \in \{E \mid c\} \longrightarrow (x \in E \wedge x \, \epsilon \, c)$$

$$y \, \epsilon \, f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$

If we do not introduce a notation for sets, but merely axioms expressing their existence, we get the axioms

$$\forall E \forall F \exists G \forall x \, (x \in G \Leftrightarrow (x = E \vee x = F))$$

$$\forall E \exists G \forall x \, (x \in G \Leftrightarrow \exists y \, (y \in E \wedge x \in y))$$

$$\forall E \exists G \forall x \, (x \in G \Leftrightarrow \forall y \, (y \in x \Rightarrow x \in E))$$

$$\forall E \forall c \exists G \forall x \, (x \in G \Leftrightarrow (x \in E \wedge x \, \epsilon \, c))$$

$$\forall x_1 \, ... \, \forall x_n \exists c \forall y \, (y \, \epsilon \, c \Leftrightarrow A)$$

A last axiom, *the extensionality axiom*, expresses when two sets are equal: when they have the same elements

$$\forall E \forall F \, ((\forall x \, (x \in E \Leftrightarrow x \in F)) \Rightarrow E = F)$$

5.6.2 Cuts

In set theory, Russell's paradox is avoided, but we can still build the class of sets that are not elements of themselves $f_{x,\neg(x\in x)}$ and the subset of any set E containing the elements of E that are not elements of themselves $C = \{E \mid f_{x,\neg x\in x}\}$ (Crabbé's set) and we can still form the proposition A expressing that this set is an element of itself $A = C \in C$. This proposition reduces to $C \in E \wedge \neg C \in C$, that is to a proposition of the form $B \wedge \neg A$. Having such a proposition A that reduces to $B \wedge \neg A$ is not contradictory, but it jeopardizes the termination of proof reduction. Indeed, the proposition $\neg B$ has the following proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B, A \vdash B \wedge \neg A}}{B, A \vdash \neg A}\quad \dfrac{}{B, A \vdash A}}{B, A \vdash \bot}}{B \vdash \neg A} \Rightarrow\text{-intro} \quad \dfrac{\dfrac{\dfrac{}{B, A \vdash B \wedge \neg A}}{B, A \vdash \neg A}\quad \dfrac{}{B, A \vdash A}}{B, A \vdash \bot}}{\dfrac{B \vdash B \quad \dfrac{}{B \vdash \neg A}}{B \vdash A}} \Rightarrow\text{-elim}$$
$$\dfrac{B \vdash \bot}{\vdash \neg B}$$

that reduces to itself in two steps. More generally, the proposition $\neg B$ has a proof, but no cut-free proof.

Thus, in this naive presentation of set theory, proof reduction does not terminate, and only recent work gave alternative formulations where proof reduction terminates.

5.6.3 Elements of mathematics

As already said, functions are not primitive objects in set theory. They are expressed as relations, that is as sets of ordered pairs. Thus, there is no comprehension scheme for functions. But nothing prevents designing a similar theory with primitive functions.

In set theory, there is usually only one base object: the empty set. Thus, unlike in arithmetic, natural numbers are not primitive objects and they need to be constructed.

One possibility is to take an axiom stating the existence of an infinite set B and to construct natural numbers as *Cantor numbers*, that is finite cardinals in B. Numbers are then elements of the powerset of the powerset of B.

Another possibility is to take an axiom stating the existence of an infinite set B, call S, *successor*, a non surjective injection from B to B, and 0, *zero*, an element that is not in the image of S, and to construct natural numbers as *Peano numbers*, that is 1 as $S(0)$, 2 as $S(S(0))$, ... Numbers are then elements of B.

A third possibility is to construct natural numbers as *Von Neumann numbers*, that is in such a way that n is the set of numbers strictly less than n. We can this way prove the existence of $0 = \emptyset$, $1 = \{\emptyset\}$, $2 = \{\emptyset, \{\emptyset\}\}$, ... But, it is necessary to have an axiom asserting the existence of the set of natural numbers.

In all cases, the set $\mathbb{N}$ must be defined as the smallest set containing 0 and closed by S so that an induction theorem can be proved.

In all cases, it is necessary to state an axiom asserting the existence of an infinite set—be it an arbitrary infinite set B or the set of natural numbers—as, otherwise, there are models of set theory, where all sets are finite.

Chapter 6

Simple type theory

Type theory has been originally proposed by Russell and then developed by Whitehead, Ramsey, Chwistek, Church, etc. leading to *Simple type theory*, also called *Higher-order logic*. The idea of *Simple type theory* is to abandon the principle that any predicate or function can be applied to any object. Objects are therefore distinguished according to their degree of functionality: base objects, propositional contents, functions mapping base objects to base objects, functions mapping functions to functions, etc. Simple type theory is thus a many-sorted theory.

6.1 Simple type theory

Unlike the two sorts of arithmetic, ι and κ, the sorts of simple type theory are an infinity, they are called *simple types*.

Definition 6.1 (Simple types) Simple types *are inductively defined as follows*

- ι *and* o *are simple types,*
- *if* A *and* B *are simple types, then* $A \to B$ *is a simple type.*

The type ι is that of base objects, o is that of propositional contents, $A \to B$ that of functions mapping objects of type A to objects of type B. We write $A_1 \to A_2 \to ... \to A_n \to B$ for the type $(A_1 \to (A_2 ... \to (A_n \to B)...))$.

The language of the theory contains an application symbol $\alpha_{A,B}$ of arity $\langle A \to B, A, B\rangle$ for each pair of types A, B, and a single predicate symbol ε of arity $\langle o\rangle$. We then could introduce for each term t whose free variables are among $x_1, ..., x_n$ a combinator $x_1, ..., x_n \mapsto t$ and for each proposition A whose free variables are among $x_1, ..., x_n$ a combinator $\{x_1, ..., x_n \mid A\}$. But, as we shall see in Propositions 6.1 and 6.2, it is sufficient to introduce combinators $K_{A,B} = x, y \mapsto x$, $S_{A,B,C} = x, y, z \mapsto (x\ z\ (y\ z))$, $\dot\top = \{|\top\}$, $\dot\bot = \{|\bot\}$, $\dot\wedge = \{x, y \mid \varepsilon(x) \wedge \varepsilon(y)\}$, $\dot\vee = \{x, y \mid \varepsilon(x) \vee \varepsilon(y)\}$, $\dot\Rightarrow = \{x, y \mid \varepsilon(x) \Rightarrow \varepsilon(y)\}$, $\dot\forall_A = \{x \mid \forall y\, \varepsilon(x\ y)\}$, and $\dot\exists_A = \{x \mid \exists y\, \varepsilon(x\ y)\}$.

Definition 6.2 (The language of Simple type theory) *The language of Simple type theory is a many-sorted language whose sorts are simple types. It contains a predicate symbol*

- ε *of arity* $\langle o \rangle$,

function symbols

- $\alpha_{A,B}$ *of arity* $\langle A \to B, A, B \rangle$,

and constants

- $K_{A,B} : A \to B \to A$,
- $S_{A,B,C} : (A \to B \to C) \to (A \to B) \to A \to C$,
- $\dot{\top} : o$,
- $\dot{\bot} : o$,
- $\dot{\wedge} : o \to o \to o$,
- $\dot{\vee} : o \to o \to o$,
- $\dot{\Rightarrow} : o \to o \to o$,
- $\dot{\forall}_A : (A \to o) \to o$,
- $\dot{\exists}_A : (A \to o) \to o$.

Simple type theory is a purely computational theory whose rules are the reduction rules associated to the combinators of the language.

Definition 6.3 (Rules of simple type theory)

$$(K_{A,B}\ x\ y) \longrightarrow x$$

$$(S_{A,B,C}\ x\ y\ z) \longrightarrow (x\ z\ (y\ z))$$

$$\varepsilon(\dot{\top}) \longrightarrow \top$$

$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$

$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow (\varepsilon(x) \vee \varepsilon(y))$$

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y))$$

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$

$$\varepsilon(\dot{\exists}_A\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

Proposition 6.1 *Let t be a term whose free variables are among $x_1, ..., x_n, y$. Then, there exists a term u whose free variables are among $x_1, ..., x_n$ such that*

$$(u\ y) \longrightarrow^* t$$

Proof. By induction on the structure of t.

- If $t = y$ then we take $u = (S\ K\ K)$ and we have $(u\ y) = (S\ K\ K\ y) \longrightarrow^1 (K\ y\ (K\ y)) \longrightarrow^1 y$.
- If $t = x_i$ then we take $u = (K\ x_i)$ and we have $(u\ y) = (K\ x_i\ y) \longrightarrow^1 x_i$.
- If t is a constant a, then we take $u = (K\ a)$ and we have $(u\ y) = (K\ a\ y) \longrightarrow^1 a$.
- If $t = (t_1\ t_2)$ then, by induction hypothesis, there exists two terms u_1 and u_2 such that $(u_1\ y) \longrightarrow^* t_1$ and $(u_2\ y) \longrightarrow^* t_2$. We take $u = (S\ u_1\ u_2)$, and we have $(u\ y) = (S\ u_1\ u_2\ y) \longrightarrow^1 ((u_1\ y)\ (u_2\ y)) \longrightarrow^* (t_1\ t_2)$.

We write this term $u = \overline{\lambda} y\ t$.

Proposition 6.2 *Let A be a proposition. Then, there exists a term u whose free variables are the same as those of A such that*

$$\varepsilon(u) \longrightarrow^* A$$

Proof. By induction on the structure of A.

- If A is atomic, it has the form $A = \varepsilon(t)$, we take $u = t$.
- If $A = \top$, we take $u = \dot{\top}$. If $A = \bot$, we take $u = \dot{\bot}$.
- If $A = A_1 \wedge A_2$ then, by induction hypothesis, there exists two terms u_1 and u_2 such that $\varepsilon(u_1) \longrightarrow^* A_1$ and $\varepsilon(u_2) \longrightarrow^* A_2$. We take $(\dot{\wedge}\ u_1\ u_2)$, and we have $\varepsilon(u) = \varepsilon(\dot{\wedge}\ u_1\ u_2) \longrightarrow^1 \varepsilon(u_1) \wedge \varepsilon(u_2) \longrightarrow^* A_1 \wedge A_2 = A$. We proceed in the same way if $A = A_1 \vee A_2$ or $A = A_1 \Rightarrow A_2$.
- If $A = \forall x\ A_1$ then, by induction hypothesis, there exists a term u such that $\varepsilon(u) \longrightarrow^* A_1$ and, by Proposition 6.1, there exists a term v such that $(v\ x) \longrightarrow^* u$. We take $(\dot{\forall}_B\ v)$, and we have $\varepsilon(u) = \varepsilon(\dot{\forall}_B\ v) \longrightarrow^1 \forall x\ \varepsilon(v\ x) \longrightarrow^* \forall x\ \varepsilon(u) \longrightarrow^* \forall x\ A_1 = A$. We proceed in the same way if $A = \exists x\ A_1$.

In Simple type theory, using a quantifier on type o, it is possible to build the proposition

$$\forall p\ (\varepsilon(p) \Rightarrow \varepsilon(p))$$

As all propositions have a propositional content (Proposition 6.2) this proposition expresses that all propositions imply themselves. In particular, it expresses that it implies itself. A theory allowing to express such partially auto-referential propositions is said to be *impredicative*. A way to avoid this impredicativity is to drop the combinators $\dot{\forall}_A$ and $\dot{\exists}_A$ when the type A contains an occurrence of the symbol o. We get this way *predicative type theory*.

Remark that the definition of Arithmetic we have given in Chapter 4 was predicative as in the symbols $f_{x_1,\ldots,x_n,y,A}$ were restricted to propositions A that do not contain the symbol ϵ. Thus, these propositions do not contain variables and quantifiers of sort κ and thus classes are build from propositions that do not contain quantification over classes. As already said, dropping this condition yields an impredicative theory: *impredicative arithmetic*.

6.2 Models of Simple type theory

Proposition 6.3 *Simple type theory has a model valued in the algebra* $\{0,1\}$. *Thus, it is consistent.*

Proof. Take $\mathcal{M}_\iota$ to be any non empty set, for instance the singleton $\{7\}$. Take $\mathcal{M}_o = \{0,1\}$. Take $\mathcal{M}_{A\to B}$ be the set of all functions from $\mathcal{M}_A$ to $\mathcal{M}_B$.

Recall that, in the algebra $\{0,1\}$, $\tilde{\top} = 1$, $\tilde{\bot} = 0$, the function $\tilde{\wedge}$, $\tilde{\vee}$, and $\tilde{\Rightarrow}$ are defined by the tables

$\tilde{\wedge}$	0	1
0	0	0
1	0	1

$\tilde{\vee}$	0	1
0	0	1
1	1	1

$\tilde{\Rightarrow}$	0	1
0	1	1
1	0	1

and the functions $\tilde{\forall}$ and $\tilde{\exists}$ are defined by the tables

$\tilde{\forall}$	$\{0\}$	$\{0,1\}$	$\{1\}$
	0	0	1

$\tilde{\exists}$	$\{0\}$	$\{0,1\}$	$\{1\}$
	0	1	1

Take $\hat{K}_{A,B}$ be the function mapping a in $\mathcal{M}_A$ and b in $\mathcal{M}_B$ to a, $\hat{S}_{A,B,C}$ be the function mapping a, b, and c to $(a\ c\ (b\ c))$, $\hat{\tilde{\top}} = \tilde{\top}$, $\hat{\tilde{\bot}} = \tilde{\bot}$, $\hat{\tilde{\wedge}} = \tilde{\wedge}$, $\hat{\tilde{\vee}} = \tilde{\vee}$, $\hat{\tilde{\Rightarrow}} = \tilde{\Rightarrow}$, $\hat{\dot{\forall}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\forall}\,\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\dot{\exists}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\exists}\,\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\alpha}_{A,B}$ be the function mapping f and a to $f(a)$, and $\hat{\varepsilon}$ be the identity.

It is routine to check that all reduction rules of Simple type theory are valid in this model. We just give one example.

$$[\![\varepsilon(\dot{\forall}_A\ x)]\!]_\phi = \tilde{\forall}\,\{\phi(x)(a) \mid a \in \mathcal{M}_A\} = [\![\forall y\ \varepsilon(x\ y)]\!]_\phi$$

Thus the rule

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow^* \forall y\ \varepsilon(x\ y)$$

is valid.

Proposition 6.4 (Super-consistency) *Simple type theory is super-consistent.*

Proof. Let $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \tilde{\forall}, \tilde{\exists}\rangle$ be a full pre-Heyting algebra—we can also assume $\mathcal{B}$ to be ordered and complete, but we shall not use this hypothesis. We build a model valued in $\mathcal{B}$ as follows. Take $\mathcal{M}_\iota$ to be any non empty set, $\mathcal{M}_o = \mathcal{B}$,

and $\mathcal{M}_{A\to B}$ be the set of all functions from $\mathcal{M}_A$ to $\mathcal{M}_B$. Let $\hat{K}_{A,B}$ be the function mapping a in $\mathcal{M}_A$ and b in $\mathcal{M}_B$ to a, $\hat{S}_{A,B,C}$ be the function mapping a, b, and c to $(a\ c\ (b\ c))$, $\hat{\top} = \tilde{\top}$, $\hat{\bot} = \tilde{\bot}$, $\hat{\wedge} = \tilde{\wedge}$, $\hat{\vee} = \tilde{\vee}$, $\hat{\Rightarrow} = \tilde{\Rightarrow}$, $\hat{\forall}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\forall}\,\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\exists}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\exists}\,\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\alpha}_{A,B}$ be the function mapping f and a to $f(a)$, and $\hat{\varepsilon}$ be the identity. It is routine to check that all reduction rules of Simple type theory are valid in this model.

6.3 Elements of mathematics

6.3.1 Equality

Equality on objects of any type A can be defined like in arithmetic. We add a constant $=_A$ of sort $A \to A \to o$ and the rule

$$x =_A y \longrightarrow (\dot{\forall}_{A\to o}\ (\overline{\lambda} c\ ((c\ x) \doteq (c\ y))))$$

where c is a variable of type $\iota \to o$, that is the equivalent in Simple type theory of the sort κ of classes. The term $x =_A y$ is a propositional content, that is a term of type o, and $\varepsilon(x =_A y)$ is a proposition. This proposition reduces to $\forall c\ (\varepsilon(c\ x) \Rightarrow \varepsilon(c\ y))$.

We sometimes omit the subscript A when it is clear form the context.

Proposition 6.5 *The proposition*

$$\forall x\ \varepsilon(x =_A x)$$

is provable. If A is an arbitrary proposition, then the proposition

$$\forall x \forall y\ (\varepsilon(x =_A y) \Rightarrow (x/z)A \Rightarrow (y/z)A)$$

is provable.

Proof. The first is equivalent to $\forall x \forall c\ (\varepsilon(c\ x) \Rightarrow \varepsilon(c\ x))$ that is provable. For the second, by Proposition 6.2, there exists a term a such that $\varepsilon(a\ z) \longrightarrow^* A$. Assume $\varepsilon(x =_A y)$, from this proposition, we deduce $\varepsilon(a\ x) \Rightarrow \varepsilon(a\ y)$ that reduces to $(x/z)A \Rightarrow (y/z)A$.

Exercise 6.1 *Consider the model of Simple type theory defined in Proposition 6.3.*

1. *Prove that $[\![\varepsilon(t =_A u)]\!]_\rho = 1$ if and only if $[\![t]\!]_\rho = [\![u]\!]_\rho$.*

   Let E be the extensionality axiom

   $$\forall f : (\iota \to \iota) \forall g : (\iota \to \iota)((\forall x : \iota\ (f\ x) =_\iota (g\ x)) \Rightarrow f =_{\iota\to\iota} g)$$

   Prove that E is valid in this model.

   Prove that $\neg E$ is not provable in Simple type theory.

2. *Build a model of Simple type theory where E is not valid.*

   Prove that E is not provable in Simple type theory.

6.3.2 Peano numbers

Like in set theory, if we want to be able to construct the set of natural numbers, we need axioms or reduction rules expressing that there is an infinite set—for instance the set of objects of type ι—, as without such an axiom, there are, as we have seen in Proposition 6.3, models of Simple type theory where all types are finite.

A way to state that there is an infinite number of base object is to state the existence of a non surjective injection f. To do so, we need a function f, its left inverse g, a set E that contains no element of the image of f, and an element a in E. This can be expressed by the reduction rules

$$(g\ (f\ x)) \longrightarrow x$$

$$(E\ a) \longrightarrow \dot{\top}$$

$$(E\ (f\ x)) \longrightarrow \dot{\bot}$$

Then, a first solution to build the natural numbers is to call S the function f, *Pred*, its left inverse, *Null* the set of elements that contains no element of the image of S and 0 an element of *Null*. This way, the rules above are reformulated

$$(\textit{Pred}\ (S\ x)) \longrightarrow x$$

$$(\textit{Null}\ 0) \longrightarrow \dot{\top}$$

$$(\textit{Null}\ (S\ x)) \longrightarrow \dot{\bot}$$

and we recognize three of the rules of Arithmetic. We can also add a rule

$$(\textit{Pred}\ 0) \longrightarrow 0$$

to make the function *Pred* total.

Then, we can define the set of natural numbers, as the smallest set containing 0 and closed by the successor function. with the rule

$$(N\ y) \longrightarrow \dot{\forall}_{\iota\to o}\overline{\lambda}c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_{\iota}\overline{\lambda}x : \iota\ ((c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))$$

but, as in Arithmetic, we shall prefer the rule

$$(N\ y) \longrightarrow \dot{\forall}_{\iota\to o}\overline{\lambda}c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_{\iota}\overline{\lambda}x : \iota\ ((N\ x) \dot{\Rightarrow} (c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))$$

Finally, it is possible to add rules for addition and multiplication, although these operations can be also defined. We get this way *Simple type theory with Peano numbers*.

Definition 6.4 (Language of Simple type theory with Peano numbers) *The language of Simple type theory with Peano numbers is a many-sorted language whose sorts are simple types. It contains the language of Simple type theory, that is a predicate symbol ε, function symbols $\alpha_{A,B}$, and constants $K_{A,B}$, $S_{A,B,C}$, $\dot{\top}$, $\dot{\bot}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\Rightarrow}$, $\dot{\forall}_A$, $\dot{\exists}_A$, $=_A$, and the symbols of arithmetic N, 0, S, Pred, $+$, $\times$, and Null.*

Definition 6.5 (Rules of simple type theory with Peano numbers) Simple type theory with Peano numbers *is the theory defined by the following reduction rules:*
(the rules of Simple type theory)

$$(K_{A,B}\ x\ y) \longrightarrow x$$

$$(S_{A,B,C}\ x\ y\ z) \longrightarrow (x\ z\ (y\ z))$$

$$\varepsilon(\dot{\top}) \longrightarrow \top$$

$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$

$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow (\varepsilon(x) \vee \varepsilon(y))$$

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y))$$

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$

$$\varepsilon(\dot{\exists}_A\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

(the rule of equality)

$$x =_A y \longrightarrow (\dot{\forall}_A\ (\overline{\lambda} c\ ((c\ x) \dot{\Rightarrow} (c\ y))))$$

(and the rules of arithmetic)

$$(N\ y) \longrightarrow \dot{\forall}_{\iota\to o}\overline{\lambda} c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_\iota \overline{\lambda} x : \iota\ ((N\ x) \dot{\Rightarrow} (c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))$$

$$(\text{Pred}\ 0) \longrightarrow 0$$

$$(\text{Pred}\ (S\ x)) \longrightarrow x$$

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow (x \times y) + y$$

$$(\text{Null}\ 0) \longrightarrow \dot{\top}$$

$$(\text{Null}\ (S\ x)) \longrightarrow \dot{\bot}$$

Proposition 6.6 Simple type theory with Peano numbers *has a model valued in the algebra* $\{0, 1\}$. *Hence it is consistent.*

Proof. We build a model valued in $\{0,1\}$ as a mixture of the model built in the proof of Propositions 4.9 and 6.3. Let $\mathcal{M}_\iota$ be, not any set, but the set $\mathbb{N}$. Let $\mathcal{M}_o = \{0,1\}$ and $\mathcal{M}_{A\to B}$ be the set of all functions from $\mathcal{M}_A$ to $\mathcal{M}_B$. Let $\hat{K}_{A,B}$ be the function mapping a in $\mathcal{M}_A$ and b in $\mathcal{M}_B$ to a, $\hat{S}_{A,B,C}$ be the function mapping a, b, and c to $(a\ c\ (b\ c))$, $\hat{\tilde{\top}} = \tilde{\top} = 1$, $\hat{\tilde{\bot}} = \tilde{\bot} = 0$, $\hat{\tilde{\wedge}} = \tilde{\wedge}$, $\hat{\tilde{\vee}} = \tilde{\vee}$, $\hat{\tilde{\Rightarrow}} = \tilde{\Rightarrow}$, $\hat{\tilde{\forall}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\forall}\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\tilde{\exists}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\exists}\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\alpha}_{A,B}$ be the function mapping f and a to $f(a)$, and $\hat{\varepsilon}$ be the identity. Let $\hat{=}_A$ be the function mapping mapping a and b to 1 if $a = b$ and to 0 otherwise. The symbols 0, S, $+$, $\times$, *Pred*, *Null* are interpreted in the obvious way and the symbol N is interpreted as the function mapping all natural numbers to 1. It is routine to check that all the rules of Simple type theory with Peano numbers are valid in this model.

Proposition 6.7 Simple type theory with Peano numbers *is super-consistent.*

Proof. Let $\mathcal{B} = \langle \mathcal{B}, \leq, \tilde{\top}, \tilde{\bot}, \tilde{\wedge}, \tilde{\vee}, \tilde{\Rightarrow}, \tilde{\forall}, \tilde{\exists}\rangle$ be a full ordered and complete pre-Heyting algebra. We build a model valued in $\mathcal{B}$ as a mixture of the model built in the proof of Propositions 4.10 and 6.4. Let $\mathcal{M}_\iota$ be, not any set, but the set $\mathbb{N}$. Let $\mathcal{M}_o = \mathcal{B}$ and $\mathcal{M}_{A\to B}$ be the set of all functions from $\mathcal{M}_A$ to $\mathcal{M}_B$. Let $\hat{K}_{A,B}$ be the function mapping a in $\mathcal{M}_A$ and b in $\mathcal{M}_B$ to a, $\hat{S}_{A,B,C}$ be the function mapping a, b, and c to $(a\ c\ (b\ c))$, $\hat{\tilde{\top}} = \tilde{\top}$, $\hat{\tilde{\bot}} = \tilde{\bot}$, $\hat{\tilde{\wedge}} = \tilde{\wedge}$, $\hat{\tilde{\vee}} = \tilde{\vee}$, $\hat{\tilde{\Rightarrow}} = \tilde{\Rightarrow}$, $\hat{\tilde{\forall}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\forall}\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\tilde{\exists}}_A$ be the function mapping the function f from $\mathcal{M}_A$ to $\mathcal{M}_o$ to $\tilde{\exists}\{f(x) \mid x \in \mathcal{M}_A\}$, $\hat{\alpha}_{A,B}$ be the function mapping f and a to $f(a)$, and $\hat{\varepsilon}$ be the identity. Let $\hat{=}_A$ be the function mapping mapping a and b to $[\![\dot{\forall}_{A\to o}\ (\overline{\lambda}c\ ((c\ x) \dot{\Rightarrow} (c\ y)))]\!]_{x=a,y=b}$ that is $\tilde{\forall}\{f(a) \tilde{\Rightarrow} f(b) \mid f \in \mathcal{M}_A \to \mathcal{B}\}$. The symbols 0, S, $+$, $\times$, *Pred* are interpreted in the obvious way. The symbol *Null* is interpreted by the function mapping 0 to $\tilde{\top}$ and the other numbers to $\tilde{\bot}$.

To define the interpretation of N, for each function f of $\mathbb{N} \to \mathcal{B}$ we can define an interpretation $\mathcal{M}_f$ of the language of the term

$$\dot{\forall}_{\iota\to o}\overline{\lambda}c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_\iota \overline{\lambda}x : \iota\ ((N\ x) \dot{\Rightarrow} (c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))$$

where the symbol N is interpreted by the function f. We define the function Φ from $\mathbb{N} \to \mathcal{B}$ to itself mapping f to the function mapping the natural number n to

$$[\![\dot{\forall}_{\iota\to o}\overline{\lambda}c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_\iota \overline{\lambda}x : \iota\ ((N\ x) \dot{\Rightarrow} (c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))]\!]^{\mathcal{M}_f}_{n/y}$$

The order on $\mathbb{N} \to \mathcal{B}$ defined by $f \sqsubseteq g$ if for all n, $f(n) \sqsubseteq g(n)$ is a complete order and the function Φ is monotone as the occurrence of N is positive in

$$\dot{\forall}_{\iota\to o}\overline{\lambda}c : \iota \to o\ ((c\ 0) \dot{\Rightarrow} \dot{\forall}_\iota \overline{\lambda}x : \iota\ ((N\ x) \dot{\Rightarrow} (c\ x) \dot{\Rightarrow} (c\ (S\ x))) \dot{\Rightarrow} (c\ y))$$

Hence it has a fixed-point g. We interpret the symbol N by the function g.

It is routine to check that all the rules of Simple type theory with Peano numbers are valid in this model.

6.3.3 Alternative definitions of natural numbers

Besides Peano numbers, natural numbers can also be defined as *Cantor numbers*, that is a as finite cardinals of elements of type ι. This way, natural numbers have type $(\iota \to o) \to o$. To do so, we first need to define the equinumerosity relation on objects of type $\iota \to o$ and then the cardinals as equivalence classes for this relation. We then define the cardinal zero and the successor function. The injectivity and non surjectivity of the successor function can be proved using the fact that there is an infinity of base objects. Finally, we define the set of natural numbers as the smallest set containing zero and closed by successor.

Finally, it is possible to define natural numbers as *Church numbers*, that is as iterators, of type $\iota \to (\iota \to \iota) \to \iota$, the number n being represented by the term $\overline{\lambda} x \overline{\lambda} f\ (f\ (f\ ...\ (f\ x)))$ with n occurrences of the symbol f. The injectivity and non surjectivity of the successor function can be proved using the fact that there is an infinity of base objects. Finally, we define the set of natural numbers as the smallest set containing zero and closed by successor.

In contrast, it is not possible to define in Simple type theory, the *Von Neumann numbers*, like in set theory: $0 = \varnothing$, $1 = \{\varnothing\}$, $2 = \{\varnothing, \{\varnothing\}\}$, etc. because the term $\{\varnothing, \{\varnothing\}\}$ is not well-typed.

6.4 The existence of functions in Simple type theory

To prove a proposition of the form $\exists x\ A$ a standard method is to prove the proposition $(t/x)A$ for some term t and then use the introduction rule of the existential quantifier. In Simple type theory, this is not always possible because, as we shall see in Chapter 7, many functions we wish to speak about cannot be expressed by a term. This is the case, for instance, of the function χ that takes the value 0 at 0 and the value 1 at other natural numbers. Thus, we cannot prove the proposition

$$\exists f\ \forall x\ (\varepsilon(N\ x) \Rightarrow ((\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((f\ x) =_\iota 0)) \wedge (\neg\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((f\ x) =_\iota 1))))$$

by simply providing a term. In fact, this proposition is not provable in Simple type theory: it is easy to build a model where it is not valid.

In contrast, it is easy to build a relation R of type $\iota \to \iota \to o$ such that the proposition $\varepsilon(R\ x\ y)$ expresses that y is the image of x by the function χ

$$R = \overline{\lambda} x \overline{\lambda} y\ ((x =_\iota 0 \mathrel{\dot{\Rightarrow}} y =_\iota 0) \mathrel{\dot{\wedge}} (\dot{\neg}(x =_\iota 0) \mathrel{\dot{\Rightarrow}} y =_\iota 1))$$

and prove, by induction on x, the proposition

$$\forall x\ (\varepsilon(N\ x) \Rightarrow \exists y\ (\varepsilon(N\ y) \wedge \varepsilon(R\ x\ y)))$$

that is

$$\forall x\ (\varepsilon(N\ x) \Rightarrow \exists y\ ((\varepsilon(x =_\iota 0) \Rightarrow \varepsilon(y =_\iota 0)) \wedge (\neg\varepsilon(x =_\iota 0) \Rightarrow \varepsilon(y =_\iota 1))))$$

To transform this relation into a genuine function of type $\iota \to \iota$, we must use a new axiom: *the axiom of choice*. To formulate this axiom, we introduce, for each type A, a

constant C_A of sort $(A \to o) \to A$, called *the choice symbol* or *Hilbert's symbol*, and then the axiom is formulated

$$\forall E\ (\exists y\ \varepsilon(E\ y)) \Rightarrow \varepsilon(E\ (C_A\ E))$$

Using this axiom, we can transform any relation into a function.

Proposition 6.8 *If R be a term of type $\iota \to \iota \to o$ and ϕ the term of $\iota \to \iota$*

$$\phi = \overline{\lambda}x\ (C_\iota\ \overline{\lambda}y\ (R\ x\ y))$$

If the proposition $\forall x\ (\varepsilon(N\ x) \Rightarrow \exists y\ \varepsilon(R\ x\ y))$ is provable then so is the proposition $\forall x\ (\varepsilon(N\ x) \Rightarrow \varepsilon(R\ x\ (\phi\ x)))$.

Proof. The sequent $\varepsilon(N\ x) \vdash \exists y\ \varepsilon(R\ x\ y)$ is provable, hence so are the sequents

$$\varepsilon(N\ x) \vdash \exists y\ \varepsilon((\overline{\lambda}y\ (R\ x\ y))\ y)$$

$$\varepsilon(N\ x) \vdash \varepsilon(\overline{\lambda}y\ (R\ x\ y))\ (C_\iota\ (\overline{\lambda}y\ (R\ x\ y)))$$

$$\varepsilon(N\ x) \vdash \varepsilon(R\ x\ (C_\iota\ (\overline{\lambda}y\ (R\ x\ y))))$$

$$\varepsilon(N\ x) \vdash \varepsilon(R\ x\ (\overline{\lambda}x\ (C_\iota\ (\overline{\lambda}y\ (R\ x\ y)))\ x))$$

that is

$$\varepsilon(N\ x) \vdash \varepsilon(R\ x\ (\phi\ x))$$

Note that there is no need for such an axiom in set theory, as, in set theory, functions are relations.

For instance, with the relation R above, the proposition $\forall x\ (\varepsilon(N\ x) \Rightarrow \exists y\ \varepsilon(R\ x\ y))$, is provable, thus we can define the function $\chi = \overline{\lambda}x\ (C_\iota\ \overline{\lambda}y\ (R\ x\ y))$, prove the proposition $\forall x\ (\varepsilon(N\ x) \Rightarrow \varepsilon(R\ x\ (\chi\ x)))$, that is

$$\forall x\ (\varepsilon(N\ x) \Rightarrow ((\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((\chi\ x) =_\iota 0)) \wedge (\neg\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((\chi\ x) =_\iota 1))))$$

and finally the proposition

$$\exists f \forall x\ (\varepsilon(N\ x) \Rightarrow ((\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((f\ x) =_\iota 0)) \wedge (\neg\varepsilon(x =_\iota 0) \Rightarrow \varepsilon((f\ x) =_\iota 1))))$$

using the introduction rule of the existential quantifier.

Using such a strong axiom to prove such a weak result may seem odd. In fact, a very weak form of the axiom of choice is needed, called the *axiom of descriptions* or sometime the *axiom of unique choice*

$$\forall E\ (\exists_1 y\ \varepsilon(E\ y)) \Rightarrow \varepsilon(E\ (C_A\ E))$$

where $\exists_1 x\ A$ is an abbreviation for $\exists x\ (A \wedge (\forall y\ (y/x)A \Rightarrow \varepsilon(y =_\iota x)))$.

Proposition 6.9 (The existence of addition) *Consider a version of Simple type theory, where the symbols $+$ and $\times$ and the reduction rules associated to these symbols are dropped. In this theory, it is possible to prove the existence of a function $plus$ such that*

$$\forall x \forall y\ (\varepsilon(N\ x) \Rightarrow \varepsilon(N\ y) \Rightarrow \varepsilon(N\ (plus\ x\ y)))$$

$$\forall y\ (\varepsilon(N\ y) \Rightarrow \varepsilon((plus\ 0\ y) =_\iota y))$$

$$\forall x \forall y\ (\varepsilon(N\ x) \Rightarrow \varepsilon(N\ y) \Rightarrow \varepsilon((plus\ (S\ x)\ y) =_\iota (S\ (plus\ x\ y))))$$

Proof. Consider the relation

$$Plus = \overline{\lambda}x\overline{\lambda}y\overline{\lambda}z\,(\dot{\forall}\overline{\lambda}r\,(\dot{\forall}\overline{\lambda}n\,(r\,0\,n\,n)\,\dot{\wedge}\,\dot{\forall}\overline{\lambda}n\,\dot{\forall}\overline{\lambda}p\,\dot{\forall}\overline{\lambda}q\,((r\,n\,p\,q)\,\dot{\Rightarrow}\,(r\,(S\,n)\,p\,(S\,q)))\,\dot{\Rightarrow}\,(r\,x\,y\,z)))$$

The natural numbers x, y, and z are related by this relation $Plus$, if they are related by all the relations r such that

$$\forall n\,\varepsilon(r\,0\,n\,n)$$

$$\forall n\forall p\forall q\,(\varepsilon(r\,n\,p\,q) \Rightarrow \varepsilon(r\,(S\,n)\,p\,(S\,q)))$$

It is easy to prove the propositions

$$\forall y\,\varepsilon(Plus\,0\,y\,y)$$

$$\forall x\forall y\forall z\,(\varepsilon(Plus\,x\,y\,z) \Rightarrow \varepsilon(Plus\,(S\,x)\,y\,(S\,z)))$$

We then prove, by induction over x, that this relation is functional, that is that for all x and y, there exists a unique z such that $\varepsilon(Plus\,x\,y\,z)$.

$$\forall x\,(\varepsilon(Nx) \Rightarrow \forall y(\varepsilon(N\,y) \Rightarrow \exists z\,(\varepsilon(N\,z)\wedge\varepsilon(Plus\,x\,y\,z)\wedge\forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,x\,y\,w) \Rightarrow \varepsilon(w =_\iota z)))))$$

We must therefore prove the propositions

$$\forall y\,(\varepsilon(N\,y) \Rightarrow \exists z\,(\varepsilon(N\,z)\wedge\varepsilon(Plus\,0\,y\,z)\wedge\forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,0\,y\,w) \Rightarrow \varepsilon(w =_\iota z))))$$

and

$$\forall y(\varepsilon(N\,y) \Rightarrow \exists z\,(\varepsilon(N\,z)\wedge\varepsilon(Plus\,(S\,x)\,y\,z)\wedge\forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,(S\,x)\,y\,w) \Rightarrow \varepsilon(w =_\iota z))))$$

under the hypothesis $(\forall y\,(\varepsilon(N\,y) \Rightarrow \exists z\,(\varepsilon(N\,z) \wedge \varepsilon(Plus\,x\,y\,z) \wedge \forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,x\,y\,w) \Rightarrow \varepsilon(w =_\iota z)))))$.

To prove the first, we prove

$$\varepsilon(N\,y) \wedge \varepsilon(Plus\,0\,y\,y) \wedge \forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,0\,y\,w) \Rightarrow \varepsilon(w =_\iota y))$$

under the hypothesis $\varepsilon(N\,y)$. The proposition $\varepsilon(N\,y)$ is an hypothesis. The proposition $\varepsilon(Plus\;0\;y\;y)$ is a consequence of $\forall y\;\varepsilon(Plus\;0\;y\;y)$. To prove $\forall w\;(\varepsilon(N\;w) \Rightarrow \varepsilon(Plus\,0\,y\,w) \Rightarrow \varepsilon(w =_\iota y))$, we prove $\varepsilon(w =_\iota y)$ under the hypotheses $\varepsilon(N\;w)$ and $\varepsilon(Plus\,0\,y\,w)$ that is

$$\forall r\,((\forall n\,\varepsilon(r\,0\,n\,n) \wedge \forall n\forall p\forall q\,\varepsilon(r\,n\,p\,q) \Rightarrow \varepsilon(r\,(S\,n)\,p\,(S\,q))) \Rightarrow \varepsilon(r\,0\,y\,w))$$

In this hypothesis, we substitute for r the term

$$\overline{\lambda}a\overline{\lambda}b\overline{\lambda}c\,((a =_\iota 0) \,\dot{\Rightarrow}\, (b =_\iota c))$$

and the result follows easily.

To prove the second, we prove

$$\varepsilon(N\,(S\,z))\wedge\varepsilon(Plus\,(S\,x)\,y\,(S\,z))\wedge\forall w\,(\varepsilon(N\,w) \Rightarrow \varepsilon(Plus\,(S\,x)\,y\,w) \Rightarrow \varepsilon(w =_\iota (S\,z)))$$

under the hypothesis $\varepsilon(N\ z) \wedge \varepsilon(Plus\ x\ y\ z) \wedge \forall w\ (\varepsilon(N\ w) \Rightarrow \varepsilon(Plus\ x\ y\ w) \Rightarrow \varepsilon(w =_\iota z))$.

The proposition $\varepsilon(N\ (S\ z))$ is a consequence of the proposition $\varepsilon(N\ z)$. The proposition $\varepsilon(Plus\ (S\ x)\ y\ (S\ z))$ is proved using $\forall x \forall y \forall z\ \varepsilon(Plus\ x\ y\ z) \Rightarrow \varepsilon(Plus\ (S\ x)\ y\ (S\ z))$. And to prove $\forall w\ (\varepsilon(N\ w) \Rightarrow \varepsilon(Plus\ (S\ x)\ y\ w) \Rightarrow \varepsilon(w =_\iota (S\ z)))$, we prove the proposition $\varepsilon(w =_\iota (S\ z))$ under the hypotheses $\varepsilon(N\ w)$ and $\varepsilon(Plus\ (S\ x)\ y\ w)$ that is

$$\forall r\ ((\forall n\ \varepsilon(r\ 0\ n\ n) \wedge \forall n \forall p \forall q\ \varepsilon(r\ n\ p\ q) \Rightarrow \varepsilon(r\ (S\ n)\ p\ (S\ q))) \Rightarrow \varepsilon(r\ (S\ x)\ y\ w))$$

In this hypothesis, we substitute for the variable r the term

$$\overline{\lambda} a \overline{\lambda} b \overline{\lambda} c\ ((Plus\ a\ b\ c)\ \dot{\wedge}\ ((a =_\iota (S\ x)\ \dot{\wedge}\ b =_\iota y) \dot{\Rightarrow} (c =_\iota (S\ z))))$$

and the result follows easily.

Using the axiom of choice or the axiom of descriptions and the proposition

$$\forall x\ (\varepsilon(Nx) \Rightarrow \forall y(\varepsilon(N\ y) \Rightarrow \exists z\ (\varepsilon(N\ z) \wedge \varepsilon(Plus\ x\ y\ z) \wedge \forall w\ (\varepsilon(N\ w) \Rightarrow \varepsilon(Plus\ x\ y\ w) \Rightarrow \varepsilon(w =_\iota z)))))$$

we get the existence of a function $plus$ such that

$$\forall x \forall y\ (\varepsilon(Nx) \Rightarrow \varepsilon(N\ y) \Rightarrow (\varepsilon(N\ (plus\ x\ y)) \wedge \varepsilon(Plus\ x\ y\ (plus\ x\ y)) \wedge \forall w\ (\varepsilon(N\ w) \Rightarrow \varepsilon(Plus\ x\ y\ w) \Rightarrow \varepsilon(w =_\iota (plus\ x\ y)))))$$

and

$$\forall x \forall y\ (\varepsilon(Nx) \Rightarrow \varepsilon(N\ y) \Rightarrow (\varepsilon(N\ (plus\ x\ y)) \wedge \forall w\ (\varepsilon(N\ w) \Rightarrow (\varepsilon(Plus\ x\ y\ w) \Leftrightarrow \varepsilon(w =_\iota (plus\ x\ y))))))$$

Thus we get

$$\forall x \forall y\ (\varepsilon(Nx) \Rightarrow \varepsilon(N\ y) \Rightarrow (\varepsilon(N\ (plus\ x\ y))))$$

From

$$\forall y\ \varepsilon(Plus\ 0\ y\ y)$$

we get

$$\forall y\ (\varepsilon(N\ y) \Rightarrow \varepsilon((plus\ 0\ y) =_\iota y))$$

and from

$$\forall x \forall y \forall z\ \varepsilon(Plus\ x\ y\ z) \Rightarrow \varepsilon(Plus\ (S\ x)\ y\ (S\ z))$$

we get

$$\forall x \forall y\ (\varepsilon(N\ x) \Rightarrow \varepsilon(N\ y) \Rightarrow \forall z\ (\varepsilon(N\ z) \Rightarrow \varepsilon((plus\ x\ y) =_\iota z) \Rightarrow \varepsilon((plus\ (S\ x)\ y) =_\iota (S\ z))))$$

and

$$\forall x \forall y\ (\varepsilon(N\ x) \Rightarrow \varepsilon(N\ y) \Rightarrow\ \varepsilon((plus\ (S\ x)\ y) =_\iota (S\ (plus\ x\ y))))$$

Remark. Two ingredients are essential in this proof of existence of the addition. The first is the axiom of choice or the axioms of descriptions that permits to transform the relation $Plus$ into the function $plus$. The second is impredicativity: the possibility to define the relation $Plus$ using a quantifier $\forall r$ on all relations, this variable r being later instantiated with terms containing the defined relation $Plus$ itself.

Remark. Although the functions χ and $plus$ are computable, the terms $(\chi\ 0)$ or $(plus\ 0\ 0)$ are provably equal to 0, but they do not reduce to 0. This motivates studying other theories, where we can build terms χ and $plus$, such that the terms $(\chi\ 0)$ and $(plus\ 0\ 0)$ are, not only provably equal to 0, but also reduce to 0.

6.5 Alternative formulations of Simple type theory

6.5.1 The λ-calculus

With the combinators S and K, for each variable x of type A and term t of type B, we have a term u of type $A \to B$, such that $(u\ x) \longrightarrow^* t$ (Proposition 6.1), but this term may be much bigger than t. Instead of having just the combinators S and K, we could take a combinator $x_1, ..., x_n \mapsto t$ for each term t. Yet, this language would still be somewhat uncomfortable as to abstract the variable y in the term $(x\ y)$ we need to use the constant $x, y \mapsto (x\ y)$ where both x and y are abstracted and apply this constant back to x. Indeed, it would make no sense to have a constant $y \mapsto (x\ y)$ where y is abstracted, but x remains free.

This leads to introduce a language where a term such as $x \mapsto t$ is not a constant, but is obtained by applying the symbol $\mapsto$ to the term t. The term $x \mapsto t$ obtained by abstracting the variable x in the term t is more often written $\lambda x : A\ t$ and this language is called *simply typed λ-calculus*.

Using such an abstraction mechanism yields a language that is more comfortable, but also more complex. In particular as the variable x free in t is bound in $\lambda x : A\ t$. The symbol λ binds a variable and we leave the realm of languages of Predicate logic.

Definition 6.6 (Simple type theory expressed with λ-calculus) Terms *are defined by the following rules*

- *variables of type A are terms of type A,*
- *the constants $\dot{\top}$ and $\dot{\bot}$ are terms of type o, the constants $\dot{\wedge}$, $\dot{\vee}$, and $\dot{\Rightarrow}$ are terms of type $o \to o \to o$, the constants $\dot{\forall}_A$ and $\dot{\exists}_A$ are terms of type $(A \to o) \to o$,*
- *if t is a term of type $A \to B$ and u a term of type A, then $\alpha_{A,B}(t, u)$, also written $(t\ u)$, is a term of type B,*
- *if t is a term of type B, and x a variable of type A then $\lambda x : A\ t$ is a term of type $A \to B$.*

Propositions *are defined by the following rules*

- *if t is a term of type o then $\varepsilon(t)$ is a proposition,*
- *$\top$ and $\bot$ are propositions,*
- *if A and B are propositions, then $A \wedge B$, $A \vee B$, and $A \Rightarrow B$ are propositions,*
- *if A is a proposition then $\forall x\ A$ and $\exists x\ B$ are propositions.*

The rules are

$$((\lambda x : A\ t)\ x) \longrightarrow t$$
$$\varepsilon(\dot{\top}) \longrightarrow \top$$
$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$
$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow (\varepsilon(x) \vee \varepsilon(y))$$
$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y))$$
$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$
$$\varepsilon(\dot{\exists}_A\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

The first rule $((\lambda x : A\ t)\ x) \longrightarrow t$ is called *the β-reduction rule*. If we do not use the same name x for the bound and the free variable, it is rephrased

$$((\lambda x : A\ t)\ u) \longrightarrow (u/x)t$$

6.5.2 Propositional contents

We have seen that in Simple type theory expressed with combinators or with λ-calculus there is a perfect correspondence between propositions and propositional contents (Proposition 6.2): each proposition can be written on the form $\varepsilon(t)$. Thus, we may want to drop the symbol ε, consider sequents of propositional contents $t_1, ..., t_n \vdash u$, and say that such as sequent is provable if the sequent $\varepsilon(t_1), ..., \varepsilon(t_n) \vdash \varepsilon(u)$ is.

Provability of sequents of propositional contents can be directly defined inductively with deduction rules such as

$$\frac{\Gamma \vdash (\dot{\wedge}\ t\ u)}{\Gamma \vdash t}$$

Chapter 7

The termination of the Simply typed λ-calculus

In Chapter 6, we have defined Simple type theory as a purely computational theory. In this chapter, we prove the termination of the reduction systems used to define this theory. This result is a corollary of a more fundamental theorem, the termination of Simply typed λ-calculus. Proving this theorem will lead us to introduce proof methods that will be used many times the rest of these course notes.

7.1 The Simply typed λ-calculus

Recall that simple types are inductively defined by two rules:

- ι and o are simple types,
- if A and B are simple types, then $A \to B$ is a simple type.

For each simple type, we consider an infinite set of variables of this type and possibly some constants. The Simply typed λ-terms are defined by

- variables and constants of type A are terms of type A,
- if t is a term of type $A \to B$ and u a term of type A, then $(t\ u)$ is a term of type B,
- if x is a variable of type A and t a term of type B, then $\lambda x : A\ t$ is a term of type $A \to B$.

The β-reduction rule is defined by

$$((\lambda x : A\ t)\ u) \longrightarrow (u/x)t$$

The notion of one-step β-reduction, confluence, termination and strong termination are defined as in Section 2.2.

Proposition 7.1 (Confluence) *Reduction is confluent in the simply typed λ-calculus.*

7.2 The termination of the Simply typed λ-calculus

The first idea to prove the termination of the Simply typed λ-calculus is to proceed by induction over term structure: variables and constants are irreducible, hence they terminate, if t terminates then $\lambda x : A\ t$ terminates—because all the reductions in $\lambda x : A\ t$ happen in t—, but the argument fails for applications: it is not the case that if two terms t and v terminate, so does the term $(t\ v)$. For instance, in the untyped λ-calculus, the term $\delta = \lambda x\ (x\ x)$ is irreducible, hence it terminates, but the term $(\delta\ \delta) = (\lambda x\ (x\ x)\ \lambda x\ (x\ x))$ reduces to itself, and thus does not terminates. The reason is that, in the term $(t\ v)$, reductions can happen in t, in v, and also at the root of the term, as in the case of $(\delta\ \delta)$. This happens when t is an abstraction $\lambda x\ u$, or when it reduces to an abstraction $\lambda x\ u$, in which case the term can reduce at the root to $(v/x)u$.

As usual, when we cannot prove a property by induction, we try to prove a stronger property: we need to prove a stronger property in the base case, we need to prove a stronger property in the inductive case, but we have a stronger induction hypothesis to help us. In our case, we shall prove that all terms of type A are elements of a set R_A whose elements strongly terminate, but moreover, if $A = B \to C$, if t is an element of $R_{B\to C}$, and t reduces to a term $\lambda x : B\ u$ then we will require that for all v in R_B, the reduct $(v/x)u$ is in R_C.

Note that in the definition of the set $R_{B\to C}$, all we need to know is the sets R_B and R_C, hence the definition of the sets R_A can be made by induction over type structure.

Definition 7.1 (The set R_A) *We define, by induction over the type A, a set of terms R_A.*

- *If $A = \iota$ or $A = o$, then a term t is an element of R_A if and only if it strongly terminates.*
- *If $A = B \to C$, then a term t is an element of R_A if and only if it strongly terminates and whenever it reduces to a term of the form $\lambda x : B\ u$, then for every term v in R_B, $(v/x)u$ is an element of R_C.*

Remark. These sets are called R_A, because their elements are sometimes called the *reducible* terms of type A. In these course notes, we use the adjective *reducible* for another notion (Definition 2.12). Hence we will avoid this terminology.

Proposition 7.2 (Variables) *Let A be a type and x a variable or a constant, then $x \in R_A$.*

Proof. By induction on the structure of A. The variable or constant x strongly terminates and it never reduces to abstraction.

Proposition 7.3 (Closure by reduction) *If t is an element of R_A and t reduces to t', then t' is an element of R_A.*

Proof. If t reduces to t' and t strongly terminates, then t' strongly terminates. Then, if $A = B \to C$ and t' reduces to a term of the form $\lambda x : B\ u$, then so does t, hence for every term v in R_B, $(v/x)u$ is an element of R_C.

Proposition 7.4 (Applications) *Let t be a term of the form $(u_1\ u_2)$ such that all the one-step reducts of t are in R_A. Then t is in R_A.*

Proof. We first prove that t strongly terminates. Let $t = t_1, t_2, ...$ be a reduction sequence issued from t. If this sequence has a single element, it is finite. Otherwise, we have $t \longrightarrow^1 t_2$, t_2 is an element of R_A, it strongly terminates, and the reduction sequence is finite.

Then, we distinguish two cases.

- If $A = \iota$ or $A = o$, then, as t strongly terminates, it is in R_A.
- If $A = B \to C$, then, as t strongly terminates, to prove that it is in R_A we need to prove that if it reduces to $\lambda x : B\ v$ then for all term w in R_B, the term $(w/x)v$ is an element of R_C. Assume t reduces to $\lambda x : B\ v$ and let $t = t_1, t_2, ..., t_n = \lambda x : B\ v$ be a reduction sequence from t to $\lambda x : B\ v$. As t_1 is an application and t_n is an abstraction, $n \geq 2$. Thus $t \longrightarrow^1 t_2 \longrightarrow^* t_n$. We have $t_2 \in R_A$, thus $(w/x)v$ is an element of R_C.

Proposition 7.5 *If $t_1 \in R_{A\to B}$ and $t_2 \in R_A$ then $(t_1\ t_2) \in R_B$.*

Proof. The term t_1 is in $R_{A\to B}$, hence it strongly terminates. Let n_1 be the maximum length of a reduction sequence issued from this term. The term t_2 is in R_A, hence it strongly terminates. Let n_2 be the maximum length of a reduction sequence issued from this term.

We prove by induction on $n_1 + n_2$ that $(t_1\ t_2)$ is in the set R_B. Using Proposition 7.4, we only need to prove that every of its one step reducts is in R_B. If the reduction takes place in t_1 or in t_2, then we apply Proposition 7.3 and the induction hypothesis. Otherwise, t_1 has the form $\lambda x : A\ u$ and the reduct is $(t_2/x)u$. By the definition of $R_{A\to B}$ this term is in R_B.

Theorem 7.1 *Let t be a term of type A and σ be a substitution mapping each variable of type B to an element of R_B, then σt is in R_A.*

Proof. By induction on the structure of t.

- If t is a variable, then, by definition of σ, σt is in R_A.
- If t is a constant, then $\sigma t = t$ is in R_A by Proposition 7.2.
- If t has the form $\lambda x : B\ u$ where u is a term of type C. We have $\sigma t = \lambda x : B\ \sigma u$, consider a reduction sequence issued from this term. This sequence can only reduce the term σu. By induction hypothesis, the term σu is an element of R_C, thus the reduction sequence is finite.

  Furthermore, every reduct of σt is of the form $\lambda x : B\ v$ where v is a reduct of σu. Let w be any term of R_B, the term $(w/x)v$ can be obtained by reduction from $((w/x) \circ \sigma)u$. By induction hypothesis, the term $((w/x) \circ \sigma)u$ is an element of R_C. Hence, by Proposition 7.3 the term $(w/x)v$ is an element of R_C. Therefore, the term $\sigma\lambda x : B\ u$ is an element of R_A.

- If the term t has the form $(u_1\ u_2)$ then u_1 is a term of type $B \to A$ and u_2 a term of type B. By induction hypothesis, the terms σu_1 and σu_2 are in the sets $R_{B\to A}$ and R_B, and by Proposition 7.5, the term $\sigma t = (\sigma u_1\ \sigma u_2)$ is an element of R_A.

Corollary 7.1 *If t is a term of type A, then t strongly terminates.*

Proof. Let t be a term of type A and σ be the substitution mapping each variable of type B to itself. Note that, by Proposition 7.2, this variable is an element of R_B. Then $t = \sigma t$ is an element of R_A. Hence it strongly terminates.

From termination and confluence, we can deduce that every term reduces to a unique irreducible term, and that the β-equivalence relation is decidable in the Simply typed λ-calculus.

7.3 Emptiness

A simple corollary of Corollary 7.1 is that if a type A contains a term whose constants and free variables are in a set Γ, then it also contains an irreducible term whose constants and free variables are in this set Γ. Indeed, if it contains a term t, it also contain its irreducible form.

The contrapositive is the following Proposition.

Proposition 7.6 (Emptyness) *Let Γ be a set of constants and variables, and A be a type. If there exists no irreducible term of type A whose constants and free variables are in Γ, then the type A is empty in Γ, that is there exists no term of type A whose constants and free variables are in Γ.*

Proof. Assume, there exists a term t of type A whose constants and free variables are in Γ. By Corollary 7.1, this term t has a irreducible form t'. The term t' is irreducible, it has type A and its constants and free variables are in Γ. A contradiction.

This gives a method to prove that a type A is empty by showing that it contains no irreducible term. To do so, we use the following Proposition.

Proposition 7.7 (Irreducible terms) *Let t be an irreducible λ-term, then t has the form $t = \lambda x_1 : A_1\ ...\ \lambda x_n : A_n\ (x\ u_1\ ...\ u_p)$ where x is a variable or a constant.*

Proof. The term t has the form $t = \lambda x_1 : A_1\ ...\ \lambda x_n : A_n\ t'$ where t' is not a λ-abstraction and n is possibly 0. The term t' has the form $(t''\ u_1\ ...\ u_p)$ where t'' is not an application and p is possibly 0. The term t'' is not an abstraction, if $p > 0$ because the term t is irreducible and if $p = 0$ because t' is not an abstraction. Thus it is a variable or a constant.

Proposition 7.8 (Atomic types) *Let P be an atomic type. There is no term of type P that contains no constants and no free variables.*

Proof. Using Proposition 7.6, we prove that there is no irreducible term t of type P that contains no constants and no free variables. By Proposition 7.7, if there existed one, it would have the form $t = \lambda x_1 : A_1 \; ... \; \lambda x_n : A_n \; (x \; u_1 \; ... \; u_p)$ where x is a variable or a constant. As the type P is atomic, we would have $n = 0$ and $t = (x \; u_1 \; ... \; u_p)$ where x is a variable or a constant, contradicting the fact that t contains no constants and no free variables.

Exercise 7.1 *Let P and Q be two atomic types.*

- *Let y a variable of type Q. Prove that there is no term of type P that contains no constants and whose only possible free variable is y.*
- *Prove that there is no term of type $(P \Rightarrow Q)$ that contains no constants and no free variables.*
- *Let y be a variable of type $(P \Rightarrow Q) \Rightarrow Q$. Prove that there is no term of type P that contains no constants and whose only possible free variable is y.*
- *Prove that there is no term of type $((P \Rightarrow Q) \Rightarrow Q) \Rightarrow P$ that contains no constants and no free variables.*
- *Let y be a variable of type $(P \Rightarrow Q) \Rightarrow P$. Prove that there is no term of type P that contain no constants and whose only possible free variable is y.*
- *Prove that there is no term of type $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$ that contains no constants and no free variables.*

7.4 Combinators

From Corollary 7.1, we can deduce several other termination results.

Recall that the language of Simply typed combinators is a many sorted-language whose sorts are simple types, containing an application symbol $\alpha_{A,B}$ of arity $\langle A \to B, A, B \rangle$ for each ordered pair of types A, B, a constant $K_{A,B}$ of sort $A \to B \to B$, for each ordered pair of types A, B and a constant $S_{A,B,C}$ of sort $(A \to B \to C) \to (A \to B) \to A \to C$ for each triple of sorts A, B, C. On these terms a notion of reduction is defined by the rules

$$(K_{A,B} \; x \; y) \longrightarrow x$$

$$(S_{A,B,C} \; x \; y \; z) \longrightarrow (x \; z \; (y \; z))$$

Proposition 7.9 *Reduction of Simply typed combinators strongly terminates.*

Proof. We define a translation from the language of combinators to Simply typed λ-calculus

- $|x| = x$,
- $|K_{A,B}| = \lambda x : A \; \lambda y : B \; x$,

- $|S_{A,B,C}| = \lambda x : A \to B \to C\ \lambda y : A \to B\ \lambda z : A\ (x\ z\ (y\ z))$,
- $|(t\ u)| = (|t|\ |u|)$.

It is then routine to check that if t and u are two terms in the language of combinators and $t \longrightarrow^1 u$, then $|t| \longrightarrow^+ |u|$. Hence if $t_1 \longrightarrow^1 t_2 \longrightarrow^1 t_3 \longrightarrow^1 ...$ is a reduction sequence in the reduction system of combinators, then $|t_1| \longrightarrow^+ |t_2| \longrightarrow^+ |t_3| \longrightarrow^+ ...$ is a subsequence of a reduction sequence in λ-calculus. Hence it is finite.

Proposition 7.10 *Reduction in the system of Definition 6.3 strongly terminates.*

Proof. Recall that this reduction system is

$$(K_{A,B}\ x\ y) \longrightarrow x$$

$$(S_{A,B,C}\ x\ y\ z) \longrightarrow (x\ z\ (y\ z))$$

$$\varepsilon(\dot{\top}) \longrightarrow \top$$

$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$

$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow (\varepsilon(x) \wedge \varepsilon(y))$$

$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow (\varepsilon(x) \vee \varepsilon(y))$$

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow (\varepsilon(x) \Rightarrow \varepsilon(y))$$

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$

$$\varepsilon(\dot{\exists}_A\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

We define a translation of the terms and the propositions of Simple type theory to Simply typed λ-calculus. In each type A, we choose a variable z_A and we let

- $|x| = x$,
- $|K_{A,B}| = \lambda x : A\ \lambda y : B\ x$,
- $|S_{A,B,C}| = \lambda x : A \to B \to C\ \lambda y : A \to B\ \lambda z : A\ (x\ z\ (y\ z))$,
- $|(t\ u)| = (|t|\ |u|)$,
- $|\dot{\top}| = |\dot{\bot}| = ((\lambda x : o\ x)\ z_o)$,
- $|\dot{\wedge}| = |\dot{\vee}| = |\dot{\Rightarrow}| = ((\lambda x : o \to o \to o\ x)\ z_{o \to o \to o})$,
- $|\dot{\forall}_A| = |\dot{\exists}_A| = \lambda x : A \to o\ (x\ z_A)$,
- $|\varepsilon(t)| = |t|$,
- $|\top| = |\bot| = z_o$,
- $|A \wedge B| = |A \vee B| = |A \Rightarrow B| = (z_{o \to o \to o}\ |A|\ |B|)$,
- $|\forall x\ A| = |\exists x\ A| = |(z_B/x)A|$ where B is the type of x.

It is then routine to check that if $t \longrightarrow^1 u$, then $|t| \longrightarrow^+ |u|$. Hence if $t_1 \longrightarrow^1 t_2 \longrightarrow^1 t_3 \longrightarrow^1 ...$ is a reduction sequence in the reduction system of combinators, then $|t_1| \longrightarrow^+ |t_2| \longrightarrow^+ |t_3| \longrightarrow^+ ...$ is a subsequence of a reduction sequence in λ-calculus. Hence it is finite.

Exercise 7.2 *Prove that the reduction system of Definition 6.6 strongly terminates.*

7.5 The computational expressivity of the simply typed λ-calculus

In Chapter 6 we have said that the function χ that takes the value 0 at 0 and the value 1 at other natural numbers cannot be expressed by a term in the Simply typed λ-calculus or in the language of combinators. We now have the tools to prove this proposition.

Definition 7.2 (Expression of a function by a term) *Let us write $\underline{n}$ for the term $(S\ (S\ ...(S\ 0)...))$ with n occurrences of the symbol S. A closed term t of type $\iota \to ... \to \iota \to \iota$ in Simply typed λ-calculus is said to* express *a function f from $\mathbb{N} \times ... \times \mathbb{N}$ to $\mathbb{N}$ if the irreducible form of $(t\ \underline{n_1}\ ...\ \underline{n_k})$ is $\underline{f(n_1, ..., n_k)}$.*

Proposition 7.11 *The functions from $\mathbb{N} \times ... \times \mathbb{N}$ to $\mathbb{N}$ that can be expressed in Simply typed λ-calculus are the constant functions and the functions that add a constant to one of their arguments.*

Proof. The constant functions are expressed by terms of the form

$$\lambda x_1 : \iota\ ...\ \lambda x_k : \iota\ (S\ (...(S\ 0)...))$$

and those that add a constant to one of their arguments by terms of the form

$$\lambda x_1 : \iota\ ...\ \lambda x_k : \iota\ (S\ (...(S\ x_i)...))$$

Conversely, let f be a function computationally expressed by the term t. This function is also expressed by the irreducible form of t, thus we can, without loss of generality, assume that the term t is irreducible. Using Proposition 7.7, the term t has the form $t = \lambda x_1 : \iota\ ...\ \lambda x_l : \iota\ (x\ u_1\ ...\ u_r)$ where x is a variable or a constant. As the term t is closed, x is either 0, S or among x_1, ..., x_l. We prove by induction over the size of t that the function f either is constant or adds a constant to one of its arguments.

- If $x = x_i$ then $r = 0$ and $t = \lambda x_1 : \iota ... \lambda x_k : \iota\ x_i$ and the function f adds zero to one of its arguments.
- If $x = 0$ then $r = 0$ and $t = \lambda x_1 : \iota ... \lambda x_k : \iota\ 0$ and the function f is constant.
- If $x = S$ then either $r = 0$ or $r = 1$
  - if $r = 1$ then $t = \lambda x_1 : \iota\ ...\ \lambda x_k : \iota\ (S\ u)$, by induction hypothesis, the term $\lambda x_1 : \iota\ ...\ \lambda x_k : \iota\ u$ expresses a function that either is constant or adds a constant to one of its arguments. Hence, the function f also.

– If $r = 0$ then $t = \lambda x_1 : \iota \ ... \ \lambda x_{k-1} : \iota \ S$ and f is the function that adds 1 to its k^{th} argument.

Corollary 7.2 *The functions from $\mathbb{N} \times ... \times \mathbb{N}$ to $\mathbb{N}$ that can be expressed in the language of combinators are the constant functions and the functions that add a constant to one of their arguments.*

Proof. The constant functions are expressed by terms of the form

$$\overline{\lambda} x_1 \ ... \ \overline{\lambda} x_k \ (S \ (...(S \ 0)...))$$

and those that add a constant to one of their arguments by terms of the form

$$\overline{\lambda} x_1 \ ... \ \overline{\lambda} x_k \ (S \ (...(S \ x_i)...))$$

Conversely, if a function is expressed by a term t in the language of combinators, it is also expressed by the term $|t|$ in the Simply typed λ-calculus.

Chapter 8

The termination of proof reduction in Predicate logic

We are now ready to prove termination of proof reduction in some theories in Deduction modulo theory. Before we address this question in Chapter 9, we focus in this chapter on termination of proof reduction in Predicate logic, that is without reduction rules.

Proving the termination of proof reduction will lead us to introduce a new language to express proofs as terms, following the Brouwer-Heyting-Kolmogorov interpretation of proofs and the Curry-de Bruijn-Howard correspondence: proofs will be expressed as terms in an extension of the Simply typed λ-calculus and the proof of the termination of proof reduction will follow the lines of the proof of termination of β-reduction in the Simply typed λ-calculus.

8.1 Proof-terms

8.1.1 Notations for derivation trees

When defining the notion of derivation tree associated to an inductively defined set, we can decide to label the nodes with the elements of the set produced by derivation rules or with the names of these rules. For instance, if we define the set of even numbers as the smallest set closed by the functions z and f where z is the function, from $\mathbb{N}^0$ to $\mathbb{N}$, $\mapsto 0$ and f is the function, from $\mathbb{N}$ to $\mathbb{N}$, $x \mapsto x + 2$, then the derivation of 6 is

$$\dfrac{\dfrac{\dfrac{\dfrac{}{0}}{2}}{4}}{6}$$

in the first case and

$$\dfrac{\dfrac{\dfrac{\overline{z}}{f}}{f}}{f}$$

in the second. We can also label each node with the pair formed with a function and its result which yields the tree

$$\begin{array}{l}\overline{0}\ z\\ \overline{2}\ f\\ \overline{4}\ f\\ \overline{6}\ f\end{array}$$

This is usually how proof-trees are written

$$\dfrac{\dfrac{}{\vdash \top}\ \top\text{-intro}\quad \dfrac{}{\vdash \top}\ \top\text{-intro}}{\vdash \top \wedge \top}\ \wedge\text{-intro}$$

But this is redundant, and we could as well, label the nodes with the rule names only. The proof above would then be written

$$\dfrac{\overline{\top\text{-intro}}\qquad \overline{\top\text{-intro}}}{\wedge\text{-intro}}$$

We can also write this tree using a linear notation, writing the tree whose root is labeled by f and and whose immediate sub-trees are written $t_1, ..., t_n$, as the term $f(t_1, ..., t_n)$. If we do so, the derivations of 6 in the set of even numbers is written $f(f(f(z)))$ and the proof above $\wedge$-intro($\top$-intro, $\top$-intro). Introducing a shorthand for the rule names and writing I for $\top$-intro and $\langle \pi_1, \pi_2 \rangle$ for $\wedge$-intro(π_1, π_2), we can write this proof-tree $\langle I, I \rangle$.

8.1.2 Contexts

Natural deduction rules permit to deduce sequents from sequents and, for instance, we have for each sequence of propositions $A_1, ..., A_n$ and for each A_i in this sequence a different axiom rule allowing to deduce the sequent $A_1, ..., A_n \vdash A_i$. Thus, the name of the axiom rule needs to be parametrized by the propositions $A_1, ..., A_n$ and A_i. For instance, the proof

$$\dfrac{\dfrac{}{P, Q \vdash P}\ \text{axiom}\quad \dfrac{}{P, Q \vdash Q}\ \text{axiom}}{P, Q \vdash P \wedge Q}\ \wedge\text{-intro}$$

cannot be written

$$\dfrac{\overline{\text{axiom}}\quad \overline{\text{axiom}}}{\wedge\text{-intro}}$$

because this tree does not contain enough information to reconstruct the proved sequent, and it needs to be written

$$\dfrac{\overline{\text{axiom}_{P,Q\vdash P}} \quad \overline{\text{axiom}_{P,Q\vdash Q}}}{\wedge\text{-intro}}$$

But it is possible to be less verbose by considering that in Natural deduction rules the conclusion of a sequent—the proposition on the right of the symbol $\vdash$— is more important than the context—the propositions on the left of this symbol. Thus Natural deduction rules in fact permit to deduce propositions from propositions and the context is just here to recall which propositions can be used in the axiom rule. We can give a name to each hypothesis of the context and use this name as name of the axiom rule. For instance, the name α can be given to the hypothesis P and the name β to the hypothesis Q. The context should then be written $\alpha : P, \beta : Q$ rather than P, Q. And, in this context, the proof above would be written

$$\dfrac{\overline{\alpha} \quad \overline{\beta}}{\wedge\text{-intro}}$$

From the context $\alpha : P, \beta : Q$, it is possible to reconstruct the proved proposition: $P \wedge Q$.

Using a linear notation and a shorthand $\langle \pi_1, \pi_2 \rangle$ for $\wedge$-intro(π_1, π_2), this proof-tree can be written $\langle \alpha, \beta \rangle$.

When a rule extends the context, such as the $\Rightarrow$-intro rule, we must indicate the name of the added hypothesis. For instance, if we decide to use the shorthand λ for the rule name $\Rightarrow$-intro, then, in the context $\alpha_1 : A_1, ..., \alpha_n : A_n$, the proof

$$\dfrac{\dfrac{\pi}{A_1, ..., A_n, B \vdash C}}{A_1, ..., A_n \vdash B \Rightarrow C} \Rightarrow\text{-intro}$$

cannot be merely written $\lambda\pi$ or $\lambda B\ \pi$, but we have to indicate the name used in π for the introduced hypothesis B, and, for instance, write this proof $\lambda\beta : B\ \pi$.

Note that the proof π is a proof in the context $\alpha_1 : A_1, ..., \alpha_n : A_n, \beta : B$ and the proof $\lambda\beta : B\ \pi$ in the context $\alpha_1 : A_1, ..., \alpha_n : A_n$. This shows that there are many similarities between hypotheses names and variables. The variable β is introduced by the symbol λ, it can be used in the proof-term π, but not elsewhere: it is bound by the symbol λ, and its scope is π. A proof in a context $\alpha_1 : A_1, ..., \alpha_n : A_n$ is thus expressed by a proof-term whose free variables are among $\alpha_1, ..., \alpha_n$.

The key operation described in the proof-reduction process in Chapter 1, where from a proof π of the sequent $\Gamma, A \vdash B$, we remove the hypothesis A in all sequents and replace the axiom rules on this proposition by a proof π' of the sequent $\Gamma \vdash A$ is just substitution: in the proof-term π, the proof-term π' is substituted for the variable α associated to A in the context Γ.

Definition 8.1 (Term notation for proofs) *A shorthand is introduced for the name of each rule of Natural deduction. The proof of a sequent whose context is $A_1, ..., A_n$, is*

expressed by a proof-term in the context $\alpha_1 : A_1, ..., \alpha_n : A_n$, where a variable has been introduced for each hypothesis. The proof-terms are expressed in a language with two sorts: one for proof-terms (written with Greek letters: π, ...) and the others for the terms of the language (written with Latin letters: t, ...).

- *The proof*

$$\frac{}{\Gamma \vdash A_i}\ \textit{axiom}$$

  is expressed by the proof-term α_i.

- *The proof*

$$\frac{}{\Gamma \vdash \top}\ \top\textit{-intro}$$

  is expressed by the proof-term I, where I is an individual symbol.

- *The proof*

$$\frac{\dfrac{\pi}{\Gamma \vdash \bot}}{\Gamma \vdash A}\ \bot\textit{-elim}$$

  is expressed by the proof-term $\delta_\bot(\pi)$, where $\delta_\bot$ is a function symbol of one argument.

- *The proof*

$$\frac{\dfrac{\pi}{\Gamma \vdash A} \quad \dfrac{\pi'}{\Gamma \vdash B}}{\Gamma \vdash A \wedge B}\ \wedge\textit{-intro}$$

  is expressed by the proof-term $\langle \pi, \pi' \rangle$, where $\langle , \rangle$ is a function symbol of two arguments.

- *The proof*

$$\frac{\dfrac{\pi}{\Gamma \vdash A \wedge B}}{\Gamma \vdash A}\ \wedge\textit{-elim}$$

  is expressed by the proof-term fst(π) and the proof

$$\frac{\dfrac{\pi}{\Gamma \vdash A \wedge B}}{\Gamma \vdash B}\ \wedge\textit{-elim}$$

  is expressed by the proof-term snd(π) where fst and snd are function symbols of one argument.

- *The proof*

$$\frac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro}$$

is expressed by the proof-term $i(\pi)$ and the proof

$$\dfrac{\dfrac{\pi}{\Gamma \vdash B}}{\Gamma \vdash A \vee B}\ \vee\textit{-intro}$$

is expressed by the proof-term $j(\pi)$, where i and j are function symbols of one argument.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash A \vee B} \quad \dfrac{\pi'}{\Gamma, A \vdash C} \quad \dfrac{\pi''}{\Gamma, B \vdash C}}{\Gamma \vdash C}\ \vee\textit{-elim}$$

is expressed by the proof-term $\delta(\pi, \alpha : A\ \pi', \beta : B\ \pi'')$, where δ is a function symbol of three arguments binding one variable in its second argument and one in its third.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\textit{-intro}$$

is expressed by the proof-term $\lambda\alpha : A\ \pi$, where λ is a function symbol binding a variable in its second argument π.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash A \Rightarrow B} \quad \dfrac{\pi'}{\Gamma \vdash A}}{\Gamma \vdash B}\ \Rightarrow\textit{-elim}$$

is expressed by the proof-term $app(\pi, \pi')$, where app is a function symbol of two arguments. This proof-term is also simply written $(\pi\ \pi')$.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash A}}{\Gamma \vdash \forall x\ A}\ \forall\textit{-intro}$$

is expressed by the proof-term $\lambda x\ \pi$, where λ is a function symbol of one argument binding one variable in its argument.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash \forall x\ A}}{\Gamma \vdash (t/x)A}\ \forall\textit{-elim}$$

is expressed by the proof-term $app(\pi, t)$ where app is a function symbol of two arguments. This proof-term is also simply written $(\pi\ t)$.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash (t/x)A}}{\Gamma \vdash \exists x\ A}\ \exists\textit{-intro}$$

is expressed by the proof-term $\langle t, \pi\rangle$ where $\langle , \rangle$ is a function symbol of two arguments.

- *The proof*

$$\dfrac{\dfrac{\pi}{\Gamma \vdash \exists x\ A} \quad \dfrac{\pi'}{\Gamma, A \vdash B}}{\Gamma \vdash B}\ \exists\textit{-elim}$$

is expressed by the proof-term $\delta_\exists(\pi, x\alpha : A\ \pi')$ where $\delta_\exists$ is a function symbol of two arguments binding two variables in its second argument.

The proof-reduction rules can now be rephrased on the proof-terms. An implication cut, for instance, has the form

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, A \vdash B}}{\Gamma \vdash A \Rightarrow B}\ \Rightarrow\text{-intro} \quad \dfrac{\pi_2}{\Gamma \vdash A}}{\Gamma \vdash B}\ \Rightarrow\text{-elim}$$

it is expressed by the proof-term $((\lambda\alpha : A\ \pi_1)\ \pi_2)$. This proof reduces to the proof obtained in this way: in the proof π_1 we suppress the hypothesis A in all sequents, then each time the axiom rule is used with this proposition, we replace it with the proof π_2. As already said, this proof is expressed by the proof-term $(\pi_2/\alpha)\pi_1$. Thus, the proof-reduction rule on proof-terms is

$$((\lambda\alpha : A\ \pi_1)\ \pi_2) \longrightarrow (\pi_2/\alpha)\pi_1$$

It is the β-reduction of λ-calculus.

Definition 8.2 (Reduction rules for proof-terms)

$$\textit{fst}(\langle \pi_1, \pi_2\rangle) \longrightarrow \pi_1$$

$$\textit{snd}(\langle \pi_1, \pi_2\rangle) \longrightarrow \pi_2$$

$$\delta(i(\pi_1), \alpha : A\ \pi_2, \beta : B\ \pi_3) \longrightarrow (\pi_1/\alpha)\pi_2$$

$$\delta(j(\pi_1), \alpha : A\ \pi_2, \beta : B\ \pi_3) \longrightarrow (\pi_1/\beta)\pi_3$$

$$((\lambda\alpha : A\ \pi_1)\ \pi_2) \longrightarrow (\pi_2/\alpha)\pi_1$$

$$((\lambda x\ \pi)\ t) \longrightarrow (t/x)\pi$$

$$\delta_\exists(\langle t, \pi_1\rangle, x\alpha : A\ \pi_2) \longrightarrow (\pi_1/\alpha)(t/x)\pi_2$$

Exercise 8.1 *What is the proof-term associated to the proof*

$$\dfrac{\dfrac{\dfrac{}{\exists x\,(P(x)\Rightarrow P(x))\vdash\exists x\,(P(x)\Rightarrow P(x))}\,\textit{axiom}}{\vdash\exists x\,(P(x)\Rightarrow P(x))\Rightarrow\exists x\,(P(x)\Rightarrow P(x))}\,\Rightarrow\textit{-intro} \qquad \dfrac{\dfrac{\dfrac{}{P(c)\vdash P(c)}\,\textit{axiom}}{\vdash P(c)\Rightarrow P(c)}\,\Rightarrow\textit{-intro}}{\vdash\exists x\,(P(x)\Rightarrow P(x))}\,\exists\textit{-intro}}{\vdash\exists x\,(P(x)\Rightarrow P(x))}\,\Rightarrow\textit{-elim}$$

? Reduce it to its irreducible form.

8.1.3 The Brouwer-Heyting-Kolmogorov interpretation

In the previous section, we have introduced a shorthand for the name for each rule of Natural deduction and written proofs as proof-terms using the usual expression of trees as terms, expressing the tree whose root is labeled by f and and whose immediate subtrees are expressed as $t_1, ..., t_n$, as the term $f(t_1, ..., t_n)$. From the number of premises of each rule we derived the number of arguments of each symbol. Then the idea of expressing a proof of a sequent $A_1, ..., A_n \vdash B$ as a proof-term whose free variables are names $\alpha_1, ..., \alpha_n$, associated to the hypotheses $A_1, ..., A_n$, gave a binding arity to each symbol.

A completely different approach, due to Brouwer, Heyting and Kolmogorov, leads to exactly the same result. This approach is based on the following remark.

To build a proof of $A \wedge B$, using the $\wedge$-intro rule, we need to build first a proof of A and a proof of B. And when we have a proof of $A \wedge B$, we can use it with the $\wedge$-elim rules, to build a proof of A or a proof of B. In the same way, to build an ordered pair formed with a proof of A and a proof of B, we need to build first a proof of A and a proof of B. And when we have an ordered pair formed with a proof of A and a proof of B, we can use it to build a proof of A or to build a proof of B. Thus, a proof of $A \wedge B$ is built and used in the same way as an ordered pair formed with a proof of A and a proof of B. In the same way, a proof of $A \Rightarrow B$ is built and used in the same way as an algorithm mapping proofs of A to proofs of B.

Thus we can say that a proof of $A \wedge B$ *is* an ordered pair formed with a proof of A and a proof of B, and that a proof of $A \Rightarrow B$ *is* an algorithm mapping proofs of A to proofs of B. More generally,

- a proof of $\top$ is always the same object,
- a proof of $\bot$, there is none,
- a proof of $A \wedge B$ is an ordered pair formed with a proof of A and a proof of B,
- a proof of $A \vee B$ is either a proof of A or a proof of B,
- a proof of $A \Rightarrow B$ is an algorithm mapping proofs of A to proofs of B,
- a proof of $\forall x\ A$ is an algorithm mapping objects t to proofs of $(t/x)A$,
- a proof of $\exists x\ A$ is an ordered pair formed with an object t and a proof of $(t/x)A$.

This way, if π is a proof of A and π' a proof of B, we write $\langle \pi_1, \pi_2 \rangle$ for the proof of $A \wedge B$ built with the $\wedge$-intro rule and if π is a proof of B using a free variable α standing for an hypothesis A, we write $\lambda\alpha : A\ \pi$ for the proof of $A \Rightarrow B$ built with the $\Rightarrow$-intro rule.

8.1.4 The Curry-de Bruijn-Howard correspondence

We now want to associate a type to each proof-term. If $\Phi(A)$ is the type of the proofs of A and $\Phi(B)$ is the type of proofs of B, then, as proofs of $A \Rightarrow B$ are algorithms mapping proofs of A to proofs of B, the type of proofs of $A \Rightarrow B$ is $\Phi(A) \to \Phi(B)$. Thus

$$\Phi(A \Rightarrow B) = \Phi(A) \to \Phi(B)$$

and Φ is an isomorphism between propositions and types: *the Curry-de Bruijn-Howard correspondence*.

Instead of using explicitly this isomorphism, we shall, as usual, identify isomorphic object, that is types and propositions, and say equivalently that $\lambda\alpha : A\ \alpha$ *has type* $A \Rightarrow A$ or *is a proof of* $A \Rightarrow A$.

For instance, the proof-term $\lambda\alpha : A\ \alpha$ has type $A \Rightarrow A$ but the proof-term $\lambda\alpha : A\ (\alpha\ \alpha)$ is not well-typed: it does not correspond to any proof.

The Natural deduction rules are now used to express which proof-terms is a proof of which proposition: they are turned into typing rules for proof-terms.

We use a notation $\alpha_1 : A_1, ..., \alpha_n : A_n \vdash \pi : B$ to express that π is a proof of the sequent $A_1, ..., A_n \vdash B$ where $\alpha_1, ..., \alpha_n$ are the names given to the variables of standing for proofs of the propositions $A_1, ..., A_n$. The rules are the following.

Definition 8.3 (Typing / Deduction rules)

$$\frac{}{\Gamma \vdash \alpha : A}\ \textit{axiom if } \alpha : A \in \Gamma$$

$$\frac{}{\Gamma \vdash I : \top}\ \top\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : \bot}{\Gamma \vdash \delta_\bot(\pi) : A}\ \bot\textit{-elim}$$

$$\frac{\Gamma \vdash \pi : A \quad \Gamma \vdash \pi' : B}{\Gamma \vdash \langle \pi, \pi' \rangle : A \wedge B}\ \wedge\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : A \wedge B}{\Gamma \vdash \textit{fst}(\pi) : A}\ \wedge\textit{-elim}$$

$$\frac{\Gamma \vdash \pi : A \wedge B}{\Gamma \vdash \textit{snd}(\pi) : B}\ \wedge\textit{-elim}$$

$$\frac{\Gamma \vdash \pi : A}{\Gamma \vdash i(\pi) : A \vee B}\ \vee\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : B}{\Gamma \vdash j(\pi) : A \vee B}\ \vee\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : A \vee B \quad \Gamma, \alpha : A \vdash \pi' : C \quad \Gamma, \beta : B \vdash \pi'' : C}{\Gamma \vdash \delta(\pi, \alpha : A\ \pi', \beta : B\ \pi'') : C}\ \vee\textit{-elim}$$

$$\frac{\Gamma, \alpha : A \vdash \pi : B}{\Gamma \vdash \lambda\alpha : A\ \pi : A \Rightarrow B}\ \Rightarrow\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : A \Rightarrow B \quad \Gamma \vdash \pi' : A}{\Gamma \vdash (\pi\ \pi') : B}\ \Rightarrow\textit{-elim}$$

$$\frac{\Gamma \vdash \pi : A}{\Gamma \vdash \lambda x\ \pi : \forall x\ A}\ \forall\textit{-intro if } x \notin FV(\Gamma)$$

$$\frac{\Gamma \vdash \pi : \forall x\ A}{\Gamma \vdash (\pi\ t) : (t/x)A}\ \forall\textit{-elim}$$

$$\frac{\Gamma \vdash \pi : (t/x)A}{\Gamma \vdash \langle t, \pi\rangle : \exists x\ A}\ \exists\textit{-intro}$$

$$\frac{\Gamma \vdash \pi : \exists x\ A \quad \Gamma, \alpha : A \vdash \pi' : B}{\Gamma \vdash \delta_\exists(\pi, x\alpha : A\ \pi') : B}\ \exists\textit{-elim if } x \notin FV(\Gamma, B)$$

Proposition 8.1 *A sequent $A_1, \ldots, A_n \vdash B$ is derivable in Natural deduction if and only if there exists a proof-term π such that $\alpha_1 : A_1, \ldots, \alpha_n : A_n \vdash \pi : B$ is derivable in this system.*

Proof. By induction over the structure of derivations.

Note that, unlike the rules of Natural deduction, that define a semi-decidable set of sequents, these typing rules define a decidable set of sequent. Indeed, in Natural deduction, when attempting to prove the sequent $\Gamma \vdash B$, one can use many rules and many instances of these rules, for instance, using the $\Rightarrow$-elim rule, one needs to try all possible A and attempt to prove the sequents $\Gamma \vdash A \Rightarrow B$ and $\Gamma \vdash A$. In contrast, with these typing rules, attempting to prove the sequent $\Gamma \vdash (\pi\ \pi') : B$, one can only use the $\Rightarrow$-elim rule. Typing the term π in the context Γ yields a type C and typing π' in this context yields a type A. All that needs to be checked is that C is equal to $A \Rightarrow B$. Thus, when applying these typing rules bottom-up, there is no choice point, no backtrack, etc. This is due to the fact that, as seen in Proposition 8.1, the derivation the sequent $\Gamma \vdash \pi : B$, that is that of the sequent $\Gamma \vdash B$ is fully contained in the proof-term π.

Proposition 8.2 (Substitution) *If $\Gamma, \alpha : A \vdash \pi : B$ and $\Gamma \vdash \pi' : A$, then $\Gamma \vdash (\pi'/\alpha)\pi : B$.*

Proof. By induction on the structure of π.

Proposition 8.3 (Subject reduction) *If $\Gamma \vdash \pi : A$ and $\pi \longrightarrow^* \pi'$ then $\Gamma \vdash \pi' : A$.*

Proof. We first prove, using Proposition 8.2, that if $\Gamma \vdash \pi : A$ and π reduces in one step at the root to π' then $\Gamma \vdash \pi' : A$. Then, we deduce by induction over term structure that if $\Gamma \vdash \pi : A$ and $\pi \longrightarrow^1 \pi'$ then $\Gamma \vdash \pi' : A$. And we conclude with an induction over the structure of reduction sequences.

8.1.5 Properties of closed irreducible proof-terms

Proposition 8.4 (Final rule) *If π is a closed and irreducible proof-term of type A, then it is an introduction, that is a term of the form I, $\langle \pi_1, \pi_2 \rangle$, $i(\pi_1)$, $j(\pi_1)$, $\lambda\alpha : B\ \pi_1$, $\lambda x\ \pi_1$, or $\langle t, \pi_1 \rangle$, but neither a variable nor an elimination, that is a term of the form $\delta_\bot(\pi_1)$, fst(π_1), snd(π_1) $\delta(\pi_1, \alpha : B\ \pi_2, \beta : C\ \pi_3)$, $(\pi_1\ \pi_2)$, $(\pi_1\ t)$, or $\delta_\exists(\pi_1, x\alpha : B\ \pi_2)$.*

Proof. By induction over the structure of π. As π is closed it cannot be a variable. If this last rule is an elimination, then the proof-term π_1, is closed and irreducible, hence, by induction hypothesis, it is an introduction and the proof-term is reducible.

Corollary 8.1 (Disjunction property, Witness property) *If π a closed and irreducible proof-term of type A in a context Γ, then*

- *if $A = \top$, then $\pi = I$,*
- *A is not $\bot$,*
- *if $A = B \wedge C$ then $\pi = \langle \pi_1, \pi_2 \rangle$, π_1 is a proof-term of type of A, and π_2 is a proof-term of type of B,*
- *if $A = B \vee C$ then $\pi = i(\pi_1)$ and π_1 is a proof-term of type of B, or $\pi = j(\pi_1)$ and π_2 is a proof-term of type of C,*
- *if $A = B \Rightarrow C$, then $\pi = \lambda\alpha : B\ \pi_1$, and π_1 is a proof-term of type C,*
- *if $A = \forall x\ B$, then $\pi = \lambda x\ \pi_1$ and and π_1 is a proof-term of type B,*
- *if $A = \exists x\ B$, then $\pi = \langle t, \pi_1 \rangle$, and π_1 is a proof-term of type $(t/x)B$.*

8.2 The termination of proof-term reduction

We are now ready to prove the termination of the reduction of proof-terms, that is that if π is a proof-term of type A is some context Γ, then π strongly terminates. This proof follows the proof of Theorem 7.1, except that instead of having one connective $\rightarrow$, we have seven.

Definition 8.4 (The set R_A) *We define by induction over the proposition A a set of proof-terms R_A.*

- *If A is an atomic proposition, then a proof-term is an element of R_A if it strongly terminates.*
- *A proof-term is an element of $R_\top$ if it strongly terminates.*
- *A proof-term is an element of $R_\bot$ if it strongly terminates.*
- *A proof-term is an element of $R_{A \wedge B}$ if it strongly terminates and when it reduces to a proof-term of the form $\langle \pi_1, \pi_2 \rangle$ then π_1 and π_2 are elements of R_A and R_B.*

- *A proof-term is an element of $R_{A \vee B}$ if it strongly terminates and when it reduces to a proof-term of the form $i(\pi_1)$ (resp. $j(\pi_2)$) then π_1 (resp. π_2) is an element of R_A (resp. R_B).*
- *A proof-term is an element of $R_{A \Rightarrow B}$ if it strongly terminates and when it reduces to a proof-term of the form $\lambda\alpha : A\ \pi_1$ then for every π' in R_A, $(\pi'/\alpha)\pi_1$ is an element of R_B.*
- *A proof-term is an element of $R_{\forall x\ A}$ if it strongly terminates and when it reduces to a proof-term of the form $\lambda x\ \pi_1$ then for every term t of sort T (where T is the sort of x), $(t/x)\pi_1$ is an element of R_A.*
- *A proof-term is an element of $R_{\exists x\ A}$ if it strongly terminates and when it reduces to a proof-term of the form $\langle t, \pi_1 \rangle$, π_1 is an element of R_A.*

Proposition 8.5 (Substitution)

$$R_{(t/x)A} = R_A$$

Proof. By induction on the structure of A.

Proposition 8.6 (Variables) *Let A be a proposition and α be a variable, then $\alpha \in R_A$.*

Proof. By induction on the structure of A. The variable α strongly terminates and it never reduces to an introduction.

Proposition 8.7 (Closure by reduction) *If π is an element of R_A and $\pi \longrightarrow^* \pi'$, then π' is an element of R_A.*

Proof. As π is an element of R_A it strongly terminates, thus π' strongly terminates. Then, if $\pi' \longrightarrow^* \pi''$, we have $\pi \longrightarrow^* \pi' \longrightarrow^* \pi''$ and we detail cases according to the form of A and π''.

- If $A = B \wedge C$ and $\pi'' = \langle \rho_1, \rho_2 \rangle$ then, as $\pi \longrightarrow^* \pi''$, $\rho_1 \in R_B$ and $\rho_2 \in R_C$.
- If $A = B \vee C$ and $\pi'' = i(\rho)$ (resp. $j(\rho)$) then, as $\pi \longrightarrow^* \pi''$, $\rho \in R_B$ (resp. $\rho \in R_C$).
- If $A = B \Rightarrow C$ and $\pi'' = \lambda\alpha : B\ \rho$ then, as $\pi \longrightarrow^* \pi''$, for all ρ' in R_B, $(\rho'/\alpha)\rho \in R_C$.
- If $A = \forall x\ B$ and $\pi'' = \lambda x\ \rho$ then, as $\pi \longrightarrow^* \pi''$, for all t of sort T (where T is the sort of x) $(t/x)\rho \in R_B$.
- If $A = \exists x\ B$ and $\pi'' = \langle t, \rho \rangle$ then, as $\pi \longrightarrow^* \pi''$, $\rho \in R_B$.

Thus $\pi' \in R_A$.

Proposition 8.8 (Eliminations) *Let A be a proposition and π a proof-term that is an elimination and such that all one-step reducts of π are in R_A, then π is in R_A.*

Proof. We first prove that π strongly terminates. Let $\pi = \pi_1, \pi_2, \ldots$ be a reduction sequence issued from π. If this sequence is empty it is finite. Otherwise, we have $\pi \longrightarrow^1 \pi_2$ and hence π_2 is an element of R_A thus it strongly terminates and the reduction sequence is finite.

Then assume that $\pi \longrightarrow^* \pi'$ where π' is an introduction and let $\pi = \pi_1, \pi_2, \ldots, \pi_n = \pi'$ be a reduction sequence from π to π'. As π is an elimination and π' is not, $n \geq 2$. Thus $\pi \longrightarrow^1 \pi_2 \longrightarrow^* \pi'$. We have $\pi_2 \in R_A$ and $\pi_2 \longrightarrow^* \pi'$, we detail cases according to the form of A and π''.

- If $A = B \wedge C$ and $\pi' = \langle \rho_1, \rho_2 \rangle$ then, as $\pi_2 \longrightarrow^* \pi'$, $\rho_1 \in R_B$ and $\rho_2 \in R_C$.
- If $A = B \vee C$ and $\pi' = i(\rho)$ (resp. $j(\rho)$) then, as $\pi_2 \longrightarrow^* \pi'$, $\rho \in R_B$ (resp. $\rho \in R_C$).
- If $A = B \Rightarrow C$ and $\pi' = \lambda\alpha : B\ \rho$ then, as $\pi_2 \longrightarrow^* \pi'$, for all ρ' in R_B, $(\rho'/\alpha)\rho \in R_C$.
- If $A = \forall x\ B$ and $\pi' = \lambda x\ \rho$ then, as $\pi_2 \longrightarrow^* \pi'$, for all t of sort T (where T is the sort of x) $(t/x)\rho \in R_B$.
- If $A = \exists x\ B$ and $\pi' = \langle t, \rho \rangle$ then, as $\pi_2 \longrightarrow^* \pi'$, $\rho \in R_B$.

Thus $\pi \in R_A$.

Theorem 8.1 *Let π be a proof-term of type A in a context Γ, θ be a substitution mapping the term-variables to terms of the same sort, and σ a substitution mapping proof-term variables bound to a proposition B in Γ to elements of R_B. Then $\sigma\theta\pi$ is an element of R_A.*

Proof. By induction over the structure of π.

- axiom. If π is a variable α, we have $(\alpha : A) \in \Gamma$. By definition $\sigma\theta\alpha = \sigma\alpha$, which is an element of R_A.
- $\top$-intro. The proposition A has the form $\top$ and the proof-term π has the form I. The proof-term $\sigma\theta\pi = I$ strongly terminates.
- $\wedge$-intro. The proposition A has the form $B \wedge C$ and the proof-term π has the form $\langle \rho_1, \rho_2 \rangle$ where ρ_1 is a proof-term of type B and ρ_2 a proof-term of type C. We have $\sigma\theta\pi = \langle \sigma\theta\rho_1, \sigma\theta\rho_2 \rangle$ and, by induction hypothesis, $\sigma\theta\rho_1 \in R_B$ and $\sigma\theta\rho_2 \in R_C$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-terms $\sigma\theta\rho_1$ and $\sigma\theta\rho_2$, thus it is finite. Furthermore, any reduct of $\sigma\theta\pi$ has the form $\langle \rho'_1, \rho'_2 \rangle$ where ρ'_1 is a reduct of $\sigma\theta\rho_1$ and ρ'_2 one of $\sigma\theta\rho_2$. By Proposition 8.7, $\rho'_1 \in R_B$ and $\rho'_2 \in R_C$.
- $\vee$-intro. The proposition A has the form $B \vee C$ and the proof-term π has the form $i(\rho)$ (resp. $j(\rho)$) where ρ is a proof-term of type of B (resp. C). We have

$\sigma\theta\pi = i(\sigma\theta\rho)$ (resp. $\sigma\theta\pi = j(\sigma\theta\rho)$) and, by induction hypothesis, $\sigma\theta\rho \in R_B$ (resp. $\sigma\theta\rho \in R_C$).

A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, all reducts of $\sigma\theta\pi$ have the form $i(\rho')$ (resp. $j(\rho')$) where ρ' is a reduct of $\sigma\theta\rho$. By Proposition 8.7, $\rho' \in R_B$ (resp. $\rho' \in R_C$).

- $\Rightarrow$-intro. The proposition A has the form $B \Rightarrow C$ and the proof-term π has the form $\lambda\alpha : B\ \rho$ where ρ is a proof-term of type C. We have $\sigma\theta\pi = \lambda\alpha : \theta B\ \sigma\theta\rho$ and, by induction hypothesis, $\sigma\theta\rho \in R_C$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, every reduct of $\sigma\theta\pi$ has the form $\lambda\alpha : \theta B\ \rho'$ where ρ' is a reduct of $\sigma\theta\rho$. Let then τ be any proof-term of R_B, the proof-term $(\tau/\alpha)\rho'$ is a reduct of $((\tau/\alpha) \circ \sigma)\theta\rho$ which, by induction hypothesis again, is an element of R_C. Thus, by Proposition 8.7, $(\tau/\alpha)\rho' \in R_C$.

- $\forall$-intro. The proposition A has the form $\forall x\ B$ and the proof-term π has the form $\lambda x\ \rho$ where ρ is a proof-term of type B. We have $\sigma\theta\pi = \lambda x\ \sigma\theta\rho$ and, by induction hypothesis, $\sigma\theta\rho \in R_B$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, all reducts of $\sigma\theta\pi$ have the form $\lambda x\ \rho'$ where ρ' is a reduct of $\sigma\theta\rho$. The proof-term $(t/x)\rho'$ is a reduct of $(t/x)\sigma\theta\rho = \sigma(t/x)\theta\rho$, which by induction hypothesis again, is an element of R_B. Thus, by Proposition 8.7, $(t/x)\rho' \in R_B$.

- $\exists$-intro. The proposition A has the form $\exists x\ B$ and and the proof-term π has the form $\langle t, \rho\rangle$ and ρ is a proof-term of type $(t/x)B$. We have $\sigma\theta\pi = \langle \theta t, \sigma\theta\rho\rangle$, and by induction hypothesis, $\sigma\theta\rho \in R_B$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, any reduct of $\sigma\theta\pi$ has the form $\langle \theta t, \rho'\rangle$ where ρ' is a reduct of $\sigma\theta\rho$. By Proposition 8.7, $\rho' \in R_B = R_{(\theta t/x)B}$.

- $\bot$-elim. The proof-term π has the form $\delta_\bot(\rho)$ where ρ is a proof-term of type $\bot$. We have $\sigma\theta\pi = \delta_\bot(\sigma\theta\rho)$, and by induction hypothesis, $\sigma\theta\rho \in R_\bot$. Hence, this proof-term strongly terminates. Let n be the maximum length of reduction sequences issued from this proof-term. We prove by induction on n that $\delta_\bot(\sigma\theta\rho) \in R_A$. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. The reduction can only take place in $\sigma\theta\rho$ and we apply the induction hypothesis.

- $\wedge$-elim. The proof-term π has the form $\mathit{fst}(\rho)$ where ρ is a proof-term of type $A \wedge B$. We have $\sigma\theta\pi = \mathit{fst}(\sigma\theta\rho)$, and by induction hypothesis, $\sigma\theta\rho \in R_{A\wedge B}$. Hence, this proof-term strongly terminates. Let n be the maximum length of a reduction sequence issued from this proof-term. We prove by induction on n that $\mathit{fst}(\sigma\theta\rho)$ is in the set R_A. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. If the reduction takes place in $\sigma\theta\rho$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho$ has the form $\langle \rho'_1, \rho'_2\rangle$ and the reduct is ρ'_1, and by the definition of $R_{A\wedge B}$ this proof-term is in R_A.

If the proof-term π has the form *snd*(ρ), the proof is the same.

- $\vee$-elim. The proof-term π has the form $(\delta\ \rho_1\ \alpha\rho_2\ \beta\rho_3)$ where ρ_1 is a proof-term of type $B \vee C$ and ρ_2 and ρ_3 are proof-terms of type A. We have $\sigma\theta\pi = (\delta\ \sigma\theta\rho_1\ \alpha\sigma\theta\rho_2\ \beta\sigma\theta\rho_3)$, and by induction hypothesis, $\sigma\theta\rho_1 \in R_{B\vee C}$, $\sigma\theta\rho_2 \in R_A$, and $\sigma\theta\rho_3 \in R_A$. Hence, these proof-terms strongly terminate. Let n, n' and n'' be the maximum length of reduction sequences issued from these proof-terms. We prove by induction on $n+n'+n''$ that $(\delta\ \sigma\theta\rho_1\ \sigma\theta\alpha\rho_2\ \beta\sigma\theta\rho_3) \in R_A$. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. If the reduction takes place in $\sigma\theta\rho_1$, $\sigma\theta\rho_2$ or $\sigma\theta\rho_3$ then we apply the induction hypothesis. Otherwise, if $\sigma\theta\rho_1$ has the form $i(\rho')$ (resp. $j(\rho')$) and the reduct is $(\rho'/\alpha)\sigma\theta\rho_2$ (resp. $(\rho'/\beta)\sigma\theta\rho_3$). By the definition of $R_{B\vee C}$ the proof-term $\rho' \in R_B$ (resp. R_C). Hence, by induction hypothesis again, $((\rho'/\alpha)\circ\sigma)\theta\rho_2 \in R_A$ (resp. $((\rho'/\beta)\circ\sigma)\theta\rho_3 \in R_A$).

- $\Rightarrow$-elim. The proof-term π has the form $(\rho_1\ \rho_2)$ and ρ_1 is a proof-term of type $B \Rightarrow A$ and ρ_2 a proof-term of type B. We have $\sigma\theta\pi = (\sigma\theta\rho_1\ \sigma\theta\rho_2)$, and by induction hypothesis $\sigma\theta\rho_1 \in R_{B\Rightarrow A}$ and $\sigma\theta\rho_2 \in R_B$. Hence these proof-terms strongly terminate. Let n be the maximum length of a reduction sequence issued from $\sigma\theta\rho_1$ and n' the maximum length of a reduction sequence issued from $\sigma\theta\rho_2$. We prove by induction on $n+n'$ that $(\sigma\theta\rho_1\ \sigma\theta\rho_2) \in R_A$. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. If the reduction takes place in $\sigma\theta\rho_1$ or in $\sigma\theta\rho_2$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho_1$ has the form $\lambda\alpha\ \rho'$ and the reduct is $(\sigma\theta\rho_2/\alpha)\rho'$. By the definition of $R_{B\Rightarrow A}$ this proof-term is in R_A.

- $\forall$-elim. The proof-term π has the form $(\rho\ t)$ where ρ is a proof-term of type $\forall x\ B$ and $A = (t/x)B$. We have $\sigma\theta\pi = (\sigma\theta\rho\ \theta t)$, and by induction hypothesis, $\sigma\theta\rho \in R_{\forall x\ B}$. Hence, this proof-term strongly terminates. Let n be the maximum length of a reduction sequence issued from this proof-term. We prove by induction on n that $(\sigma\theta\rho\ \theta t)$ is in the set R_A. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. If the reduction takes place in $\sigma\theta\rho$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho$ has the form $\lambda x\ \rho'$ and the reduct is in $(\theta t/x)\rho'$. By the definition of $R_{\forall x\ B}$, this proof-term is in $R_B = R_{(t/x)B} = R_A$.

- $\exists$-elim. The proof-term π has the form $\delta_\exists(\rho_1, x\alpha\rho_2)$ where ρ_1 is a proof-term of type $\exists x\ B$ and ρ_2 is a proof-term of type A. We have $\sigma\theta\pi = \delta_\exists(\sigma\theta\rho_1, x\alpha\sigma\theta\rho_2)$ and by induction hypothesis, $\sigma\theta\rho_1 \in R_{\exists x\ B}$ and $\sigma\theta\rho_2 \in R_A$. Hence, these proof-terms strongly terminate. Let n and n' be the maximum length of reduction sequences issued from these proof-terms. We prove by induction on $n+n'$ that $\delta_\exists(\sigma\theta\rho_1, x\alpha\sigma\theta\rho_2) \in R_A$. Using Proposition 8.8, we only need to prove that every of its one step reducts is in R_A. If the reduction takes place in $\sigma\theta\rho_1$ or $\sigma\theta\rho_2$ then we apply the induction hypothesis. Otherwise, if $\sigma\theta\rho_1$ has the form $\langle t, \rho'\rangle$ and the reduct is $(\rho'/\alpha)(t/x)\sigma\theta\rho_2 = ((\rho'/\alpha)\circ(t/x)\sigma)((t/x)\circ\theta)\rho_2$. By the definition of $R_{\exists x\ B}$, $\rho' \in R_B$. Thus, by induction hypothesis again, $((\rho'/\alpha)\circ(t/x)\sigma)((t/x)\circ\theta)\rho_2 \in R_A$.

Corollary 8.2 *Every proof-term in Predicate logic strongly terminates.*

Proof. Let π be a proof-term of type A in a context Γ. Let θ be the substitution mapping each term variable to itself, and σ the substitution mapping each proof-term variables bound to a proposition B in Γ to itself. Note that, by Proposition 8.6, this variable is an element of the set R_B. Then $\pi = \sigma\theta\pi$ is an element of R_A. Hence it strongly terminates.

Chapter 9

The termination of proof-term reduction in Deduction modulo theory

We now want to extend this termination result (Corollary 8.2) to Deduction modulo theory. We have seen that, in some theories, proof reduction does not terminate, for instance in the theory defined by the reduction rule $P \longrightarrow P \Rightarrow Q$ (Chapter 2) and in the theory defined by the rule $P \longrightarrow Q \wedge \neg P$ (Chapter 5).

In contrast, it is not difficult to prove that the proof reduction terminates for the theory defined by the reduction system $P \longrightarrow Q \Rightarrow Q$. Indeed, P can be just considered as a abbreviation for $Q \Rightarrow Q$ and all we need to do is to mimic the proof presented in Chapter 8 letting $R_P = R_{Q \Rightarrow Q}$ instead of letting it be the set of all strongly terminating proof-terms. Thus, to extend the proof of Chapter 8 to Deduction modulo theory, the key idea is to associate a set of proof-terms to each proposition in such as way that if $A \equiv B$ then $R_A = R_B$.

The sets of proof-terms that are candidate to be associated to propositions are called *candidates*, or sometimes *reducibility candidates*.

9.1 Candidates

In Chapter 8, we have defined the set $R_{A \wedge B}$ as the set of proof-terms π, such that π strongly terminates and if π reduces to a proof-term of the form $\langle \pi_1, \pi_2 \rangle$ then π_1 is an element of R_A and π_2 is an element of R_B. We can give the name $\tilde{\wedge}$ to this operation and, given two sets of proof-terms E and F, define the set of proof-terms $E \mathbin{\tilde{\wedge}} F$ as the set of proof-terms π, such that π strongly terminates and if π reduces to a proof-term of the form $\langle \pi_1, \pi_2 \rangle$ then π_1 is an element of E and π_2 is an element of F. Then, using this operation, we can define the set $R_{A \wedge B}$ as $R_A \mathbin{\tilde{\wedge}} R_B$. This leads to define the following operations.

Definition 9.1 (Operations on sets of proof-terms)

- *The set $\tilde{\top}$ is the set of strongly terminating proof-terms.*
- *The set $\tilde{\bot}$ also is the set of strongly terminating proof-terms.*
- *The function $\tilde{\wedge}$ maps the set of proof-terms E and F to the set of proof-terms π such that π strongly terminates and if π reduces to a proof-term of the form $\langle \pi_1, \pi_2 \rangle$ then $\pi_1 \in E$ and $\pi_2 \in F$.*
- *The function $\tilde{\vee}$ maps the set of proof-terms E and F to the set of proof-terms π such that π strongly terminates and when it reduces to a proof-term of the form $i(\pi_1)$ (resp. $j(\pi_2)$) then $\pi_1 \in E$ (resp. $\pi_2 \in F$).*
- *The function $\tilde{\Rightarrow}$ maps the set of proof-terms E and F to the set of proof-terms π such that π strongly terminates and if π reduces to a proof-term of the form $\lambda\alpha : A\ \pi_1$ then for every π' in E, $(\pi'/\alpha)\pi_1 \in F$.*
- *The function $\tilde{\forall}$ maps the set of sets of proof-terms S to the set of proof-terms π such that π strongly terminates and if π reduces to a proof-term of the form $\lambda x\ \pi_1$ then for every term t of sort T (where T is the sort of x), $(t/x)\pi_1$ is an element of all the elements of S.*
- *The function $\tilde{\exists}$ maps the set of sets of proof-terms S to the set of proof-terms π such that π strongly terminates and if π reduces to a proof-term of the form $\langle t, \pi_1 \rangle$, π_1 is an element of some element of S.*

The set of *candidates* can be inductively defined as the smallest set closed by these operations and intersection.

Definition 9.2 (Candidate) *The set of* Candidates *is inductively defined follows:*

- *the set of proof-terms $\tilde{\top}$ and $\tilde{\bot}$ are candidates,*
- *if E and F are candidates, then the sets of proof-terms $E\ \tilde{\wedge}\ F$, $E\ \tilde{\vee}\ F$, and $E\ \tilde{\Rightarrow}\ F$ are candidates,*
- *if S is a set of candidates, then $\tilde{\forall}\ S$ and $\tilde{\exists}\ S$ are candidates,*
- *if S is a set of candidates, then $\bigcap S$ is a candidate.*

We write $\mathcal{C}$ for the set of candidates.

Note that, in Definition 8.4, all the sets R_A are candidates.

The notion of *candidate* is due to Girard and the idea to define candidates inductively to Parigot.

9.2 The algebra of candidates

Consider a theory in Deduction modulo theory, if we can define a function R mapping every proposition A to a candidate R_A in such a way that

1. $R_{A\wedge B} = R_A \tilde{\wedge} R_B$, $R_{A\vee B} = R_A \tilde{\vee} R_B$, etc.

2. if $A \equiv B$, then $R_A = R_B$.

then we will be able to mimic the proof of Chapter 8 and prove that all proof-terms of type A are in R_A, hence strongly terminate.

Because the function R must verify the condition (1.) above, once it is defined on atomic propositions, it can be extended to all propositions in a unique way. Thus, all we have to do is to define a function mapping atomic propositions to candidates. To do so, we can define, for each predicate symbol P of arity n, a function $\overline{P}$ mapping n-tuples of terms to candidates and define $R_{P(t_1,...,t_n)}$ as $\overline{P}(t_1, ..., t_n)$. Finally, to define this function we can, in a first step associate, to each term t, an element of an arbitrary set $\mathcal{M}$ and then define a function $\hat{P}$ from $\mathcal{M}^n$ to $\mathcal{C}$. It should be clear that this construction is just the construction of a model valued in the algebra of candidates and that the set R_A is just the interpretation $[\![A]\!]$ of the proposition A.

The condition (2.) above rephrases: if $A \equiv B$ then $[\![A]\!] = [\![B]\!]$. It expresses that the model is a model of the congruence.

It is straightforward to prove that the set $\mathcal{C}$ equipped with the trivial pre-order relation $\leq$ defined by $C \leq C'$ always, and with the operations $\tilde{\top}$, $\tilde{\bot}$, $\tilde{\wedge}$, $\tilde{\vee}$, $\tilde{\Rightarrow}$, $\tilde{\forall}$ and $\tilde{\exists}$ is a pre-Heyting algebra, as any set equipped with this trivial pre-order relation and any set of operations is a pre-Heyting algebra. This pre-Heyting algebra is full as the functions $\tilde{\forall}$ and $\tilde{\exists}$ are defined for any set S.

Note that the algebra of candidates is not a Heyting algebra whatever the pre-order relation is, because $\tilde{\top}$ and $\tilde{\top} \tilde{\Rightarrow} \tilde{\top}$ are always the same element in a Heyting algebra, but they are distinct candidates. Indeed, the proof-term $\lambda\alpha : A\ (\alpha\ \alpha)$ is irreducible, hence it strongly terminates, thus it is an element of $\tilde{\top}$, but it is not an element of $\tilde{\top} \tilde{\Rightarrow} \tilde{\top}$, because $(\lambda\beta : A\ (\beta\ \beta)/\alpha)(\alpha\ \alpha)$ does not not strongly terminate. Being able to consider models valued in the algebra of candidates is another motivations for dropping antisymmetry.

Finally, the algebra of candidates can be ordered with the subset relation and, for this order, it is complete, as any set S of candidate has a greatest lowerbound $\bigcap S$. Making this algebra complete is the only reason to include closure by intersection in Definition 9.2.

9.3 The termination of proof-term reduction

Proposition 9.1 (Termination) *If C is a candidate, then all the elements of C strongly terminate.*

Proof. By induction on the construction of C.

Proposition 9.2 (Variables) *Let C be a candidate and α a variable, then $\alpha \in C$.*

Proof. By induction on the construction of C. The variable α strongly terminates and it never reduces to an introduction.

Proposition 9.3 (Closure by reduction) *If C is a candidate, π is an element of C and $\pi \longrightarrow^* \pi'$, then π' is an element of C.*

Proof. By induction on the construction of C.

If C is the intersection of a set of candidates S, then π is an element of all the elements of S, thus, by induction hypothesis, π' is an all the element of S, hence it is an element of C.

Otherwise, as π is an element of C it strongly terminates, thus π' strongly terminates. Then, if $\pi' \longrightarrow^* \pi''$, $\pi \longrightarrow^* \pi' \longrightarrow^* \pi''$, we detail cases according to the rule used to construct C and to the form of π''.

- If $C = E \tilde{\wedge} F$ and $\pi'' = \langle \rho_1, \rho_2 \rangle$ then, as $\pi \longrightarrow^* \pi''$, $\rho_1 \in E$ and $\rho_2 \in F$.
- If $C = E \tilde{\vee} F$ and $\pi'' = i(\rho)$ (resp. $j(\rho)$) then, as $\pi \longrightarrow^* \pi''$, $\rho \in E$ (resp. $\rho \in F$).
- If $C = E \tilde{\Rightarrow} F$ and $\pi'' = \lambda\alpha : B\ \rho$ then, as $\pi \longrightarrow^* \pi''$, for all ρ' in E, $(\rho'/\alpha)\rho \in F$.
- If $C = \tilde{\forall} S$ and $\pi'' = \lambda x\ \rho$ then, as $\pi \longrightarrow^* \pi''$, for all t of sort T (where T is the sort of x) $(t/x)\rho$ is an element of all the elements of S.
- If $A = \tilde{\exists} S$ and $\pi'' = \langle t, \rho \rangle$ then, as $\pi \longrightarrow^* \pi''$, ρ is an element of some element of S.

Thus $\pi' \in C$.

Proposition 9.4 (Eliminations) *Let C be a candidate and π a proof-term that is an elimination and such that all one-step reducts of π are in C, then π is in C.*

Proof. By induction on the construction of C.

If C is the intersection of a set of candidates S, then all the one step reducts of π are elements of all the elements of S, thus, by induction hypothesis, π is an element of all the element of S, hence it is an element of C.

Otherwise, we first prove that π strongly terminates. Let $\pi = \pi_1, \pi_2, \ldots$ be a reduction sequence issued from π. If this sequence is empty it is finite. Otherwise, we have $\pi \longrightarrow^1 \pi_2$ and hence π_2 is an element of C thus it strongly terminates and the reduction sequence is finite.

Then assume that $\pi \longrightarrow^* \pi'$ where π' is an introduction and let $\pi = \pi_1, \pi_2, \ldots, \pi_n = \pi'$ be a reduction sequence from π to π'. As π is an elimination and π' is not, $n \geq 2$. Thus $\pi \longrightarrow^1 \pi_2 \longrightarrow^* \pi'$. We have $\pi_2 \in C$ and $\pi_2 \longrightarrow^* \pi'$, we detail cases according to the rule used to construct C and to the form of π'.

- If $C = E \tilde{\wedge} F$ and $\pi' = \langle \rho_1, \rho_2 \rangle$ then, as $\pi_2 \longrightarrow^* \pi'$, $\rho_1 \in E$ and $\rho_2 \in F$.
- If $C = E \tilde{\vee} F$ and $\pi' = i(\rho)$ (resp. $j(\rho)$) then, as $\pi_2 \longrightarrow^* \pi'$, $\rho \in E$ (resp. $\rho \in F$).
- If $C = E \tilde{\Rightarrow} F$ and $\pi' = \lambda\alpha : B\ \rho$ then, as $\pi_2 \longrightarrow^* \pi'$, for all ρ' in E, $(\rho'/\alpha)\rho \in F$.

- If $C = \tilde{\forall}\ S$ and $\pi' = \lambda x\ \rho$ then, as $\pi_2 \longrightarrow^* \pi'$, for all t of sort T (where T is the sort of x) $(t/x)\rho$ is an element of all the elements of S.
- If $A = \tilde{\exists}\ S$ and $\pi' = \langle t, \rho\rangle$ then, as $\pi_2 \longrightarrow^* \pi'$, ρ is in some element of S.

Thus $\pi \in C$.

Theorem 9.1 *Let $\equiv$ be a congruence, $\mathcal{M}$ be a model valued in the algebra $\mathcal{C}$ of $\equiv$, π be a proof-term of type A in a context Γ, θ be a substitution mapping the variables of any sort to terms of the same sort, ϕ be a valuation mapping variables of any sort T to elements of $\mathcal{M}_T$, and σ be a substitution mapping any proof-term variable bound to proposition B in Γ to an element of $[\![B]\!]_\phi$.*

Then $\sigma\theta\pi$ is an element of $[\![A]\!]_\phi$.

Proof. By induction over the structure of π.

- axiom. If π is a variable α, we have $(\alpha : B) \in \Gamma$ with $A \equiv B$. By definition $\sigma\theta\alpha = \sigma\alpha$, which is an element of $[\![B]\!]_\phi = [\![A]\!]_\phi$.
- $\top$-intro. The proposition A is congruent to $\top$ and the proof-term π has the form I. We have $[\![A]\!]_\phi = [\![\top]\!]_\phi$, we prove $\sigma\theta\pi \in [\![\top]\!]_\phi$.

  The proof-term $\sigma\theta\pi = I$ strongly terminates.
- $\wedge$-intro. The proposition A is congruent to $B \wedge C$ and the proof-term π has the form $\langle\rho_1, \rho_2\rangle$ where ρ_1 is a proof-term of type B and ρ_2 a proof-term of type C. We have $[\![A]\!]_\phi = [\![B \wedge C]\!]_\phi$, we prove $\sigma\theta\pi \in [\![B \wedge C]\!]_\phi$.

  We have $\sigma\theta\pi = \langle\sigma\theta\rho_1, \sigma\theta\rho_2\rangle$ and, by induction hypothesis, $\sigma\theta\rho_1 \in [\![B]\!]_\phi$ and $\sigma\theta\rho_2 \in [\![C]\!]_\phi$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-terms $\sigma\theta\rho_1$ and $\sigma\theta\rho_2$, thus it is finite. Furthermore, any reduct of $\sigma\theta\pi$ has the form $\langle\rho_1', \rho_2'\rangle$ where ρ_1' is a reduct of $\sigma\theta\rho_1$ and ρ_2' one of $\sigma\theta\rho_2$. By Proposition 9.3, $\rho_1' \in [\![B]\!]_\phi$ and $\rho_2' \in [\![C]\!]_\phi$.
- $\vee$-intro. The proposition A is congruent to $B \vee C$ and the proof-term π has the form $i(\rho)$ (resp. $j(\rho)$) where ρ is a proof-term of type B (resp. C). We have $[\![A]\!]_\phi = [\![B \vee C]\!]_\phi$, we prove $\sigma\theta\pi \in [\![B \vee C]\!]_\phi$.

  We have $\sigma\theta\pi = i(\sigma\theta\rho)$ (resp. $\sigma\theta\pi = j(\sigma\theta\rho)$) and, by induction hypothesis, $\sigma\theta\rho \in [\![B]\!]_\phi$ (resp. $\sigma\theta\rho \in [\![C]\!]_\phi$).

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, all reducts of $\sigma\theta\pi$ have the form $i(\rho')$ (resp. $j(\rho')$) where ρ' is a reduct of $\sigma\theta\rho$. By Proposition 9.3, $\rho' \in [\![B]\!]_\phi$ (resp. $\rho' \in [\![C]\!]_\phi$).
- $\Rightarrow$-intro. The proposition A is congruent to $B \Rightarrow C$ and the proof-term π has the form $\lambda\alpha : B\ \rho$ where where ρ a proof-term of type C. We have $[\![A]\!]_\phi = [\![B \Rightarrow C]\!]_\phi$, we prove $\sigma\theta\pi \in [\![B \Rightarrow C]\!]_\phi$.

  We have $\sigma\theta\pi = \lambda\alpha : \theta B\ \sigma\theta\rho$ and, by induction hypothesis, $\sigma\theta\rho \in [\![C]\!]_\phi$.

A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, every reduct of $\sigma\theta\pi$ has the form $\lambda\alpha : \theta B\ \rho'$ where ρ' is a reduct of $\sigma\theta\rho$. Let then τ be any proof-term of $[\![B]\!]_\phi$, the proof-term $(\tau/\alpha)\rho'$ is a reduct of $((\tau/\alpha) \circ \sigma)\theta\rho$ which, by induction hypothesis again, is an element of $[\![C]\!]_\phi$. Thus, by Proposition 9.3, $(\tau/\alpha)\rho' \in [\![C]\!]_\phi$.

- $\forall$-intro. The proposition A is congruent to $\forall x\ B$ and the proof-term π has the form $\lambda x\ \rho$ where ρ is a proof-term of type B. We have $[\![A]\!]_\phi = [\![\forall x\ B]\!]_\phi$, we prove $\sigma\theta\pi \in [\![\forall x\ B]\!]_\phi$.

  We have $\sigma\theta\pi = \lambda x\ \sigma\theta\rho$ and, by induction hypothesis, $\sigma\theta\rho \in [\![B]\!]_{\phi,x=E}$ for all E in $\mathcal{M}_T$ (where T is the sort of x).

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, all reducts of $\sigma\theta\pi$ have the form $\lambda x\ \rho'$ where ρ' is a reduct of $\sigma\theta\rho$. The proof-term $(t/x)\rho'$ is a reduct of $(t/x)\sigma\theta\rho = \sigma(t/x)\theta\rho$, which by induction hypothesis again, is an element of $[\![B]\!]_{\phi,x=E}$ for all E in $\mathcal{M}_T$. Thus, by Proposition 9.3, $(t/x)\rho' \in [\![B]\!]_{\phi,x=E}$ for all E in $\mathcal{M}_T$.

- $\exists$-intro. The proposition A is congruent to $\exists x\ B$ and and the proof-term π has the form $\langle t, \rho\rangle$ and ρ is a proof-term of type $(t/x)B$. We have $[\![A]\!]_\phi = [\![\exists x\ B]\!]_\phi$, we prove $\sigma\theta\pi \in [\![\exists x\ B]\!]_\phi$.

  We have $\sigma\theta\pi = \langle\theta t, \sigma\theta\rho\rangle$, and by induction hypothesis, there exists an element E of $\mathcal{M}_T$ (where T is the sort of x) such that $\sigma\theta\rho \in [\![B]\!]_{\phi,x=E}$.

  A reduction sequence issued from $\sigma\theta\pi$ can only reduce the proof-term $\sigma\theta\rho$, thus it is finite. Furthermore, any reduct of $\sigma\theta\pi$ has the form $\langle\theta t, \rho'\rangle$ where ρ' is a reduct of $\sigma\theta\rho$. By Proposition 9.3, $\rho' \in [\![B]\!]_{\phi,x=E}$.

- $\bot$-elim. The proof-term π is congruent to $\delta_\bot(\rho)$ where ρ is a proof-term of type $\bot$. We have $\sigma\theta\pi = \delta_\bot(\sigma\theta\rho)$, and by induction hypothesis, $\sigma\theta\rho \in [\![\bot]\!]_\phi$. Hence, this proof-term strongly terminates. Let n be the maximum length of reduction sequences issued from this proof-term. We prove by induction on n that $\delta_\bot(\sigma\theta\rho) \in [\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![A]\!]_\phi$. The reduction can only take place in $\sigma\theta\rho$ and we apply the induction hypothesis.

- $\wedge$-elim. The proof-term π is congruent to $\mathit{fst}(\rho)$ where ρ is a proof-term of type $A \wedge B$. We have $\sigma\theta\pi = \mathit{fst}(\sigma\theta\rho)$, and by induction hypothesis, $\sigma\theta\rho \in [\![A \wedge B]\!]_\phi$. Hence, this proof-term strongly terminates. Let n be the maximum length of a reduction sequence issued from this proof-term. We prove by induction on n that $\mathit{fst}(\sigma\theta\rho)$ is in the set $[\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![B]\!]_\phi$. If the reduction takes place in $\sigma\theta\rho$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho$ has the form $\langle\rho'_1, \rho'_2\rangle$ and the reduct is ρ'_1, and by the definition of $[\![A \wedge B]\!]_\phi$ this proof-term is in $[\![A]\!]_\phi$.

  If the proof-term π has the form $\mathit{snd}(\rho)$, the proof is the same.

- $\vee$-elim. The proof-term π has the form $(\delta\ \rho_1\ \alpha\rho_2\ \beta\rho_3)$ where ρ_1 is a proof-term of type $B \vee C$ and ρ_2 and ρ_3 are proof-terms of type A. We have $\sigma\theta\pi =$

$(\delta\ \sigma\theta\rho_1\ \alpha\sigma\theta\rho_2\ \beta\sigma\theta\rho_3)$, and by induction hypothesis, $\sigma\theta\rho_1 \in [\![B \vee C]\!]_\phi$, $\sigma\theta\rho_2 \in [\![A]\!]_\phi$, and $\sigma\theta\rho_3 \in [\![A]\!]_\phi$. Hence, these proof-terms strongly terminate. Let n, n' and n'' be the maximum length of reduction sequences issued from these proof-terms. We prove by induction on $n+n'+n''$ that $(\delta\ \sigma\theta\rho_1\ \sigma\theta\alpha\rho_2\ \beta\sigma\theta\rho_3) \in [\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![A]\!]_\phi$. If the reduction takes place in $\sigma\theta\rho_1$, $\sigma\theta\rho_2$ or $\sigma\theta\rho_3$ then we apply the induction hypothesis. Otherwise, if $\sigma\theta\rho_1$ has the form $i(\rho')$ (resp. $j(\rho')$) and the reduct is $(\rho'/\alpha)\sigma\theta\rho_2$ (resp. $(\rho'/\beta)\sigma\theta\rho_3$). By the definition of $[\![B \vee C]\!]_\phi$ the proof-term $\rho' \in [\![B]\!]_\phi$ (resp. $[\![C]\!]_\phi$). Hence, by induction hypothesis again, $((\rho'/\alpha) \circ \sigma)\theta\rho_2 \in [\![A]\!]_\phi$ (resp. $((\rho'/\beta) \circ \sigma)\theta\rho_3 \in [\![A]\!]_\phi$).

- $\Rightarrow$-elim. The proof-term π has the form $(\rho_1\ \rho_2)$ and ρ_1 is a proof-term of type $B \Rightarrow A$ and ρ_2 a proof-term of type B. We have $\sigma\theta\pi = (\sigma\theta\rho_1\ \sigma\theta\rho_2)$, and by induction hypothesis $\sigma\theta\rho_1 \in [\![B \Rightarrow A]\!]_\phi$ and $\sigma\theta\rho_2 \in [\![B]\!]_\phi$. Hence these proof-terms strongly terminate. Let n be the maximum length of a reduction sequence issued from $\sigma\theta\rho_1$ and n' the maximum length of a reduction sequence issued from $\sigma\theta\rho_2$. We prove by induction on $n + n'$ that $(\sigma\theta\rho_1\ \sigma\theta\rho_2) \in [\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![A]\!]_\phi$. If the reduction takes place in $\sigma\theta\rho_1$ or in $\sigma\theta\rho_2$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho_1$ has the form $\lambda\alpha\ \rho'$ and the reduct is $(\sigma\theta\rho_2/\alpha)\rho'$. By the definition of $[\![B \Rightarrow A]\!]_\phi$ this proof-term is in $[\![A]\!]_\phi$.

- $\forall$-elim. The proof-term π has the form $(\rho\ t)$ where ρ is a proof-term of type $\forall x\ B$ and $A \equiv (t/x)B$. We have $\sigma\theta\pi = (\sigma\theta\rho\ \theta t)$, and by induction hypothesis, $\sigma\theta\rho \in [\![\forall x\ B]\!]_\phi$. Hence, this proof-term strongly terminates. Let n be the maximum length of a reduction sequence issued from this proof-term. We prove by induction on n that $(\sigma\theta\rho\ \theta t)$ is in the set $[\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![A]\!]_\phi$. If the reduction takes place in $\sigma\theta\rho$ then we apply the induction hypothesis. Otherwise, $\sigma\theta\rho$ has the form $\lambda x\ \rho'$ and the reduct is in $(\theta t/x)\rho'$. By the definition of $[\![\forall x\ B]\!]_\phi$, this proof-term is in $[\![B]\!]_{\phi,x=[\![t]\!]_\phi} = [\![(t/x)B]\!]_\phi = [\![A]\!]_\phi$.

- $\exists$-elim. The proof-term π has the form $\delta_\exists(\rho_1, x\alpha\rho_2)$ where ρ_1 is a proof-term of type $\exists x\ B$ and ρ_2 is a proof-term of type A. We have $\sigma\theta\pi = \delta_\exists(\sigma\theta\rho_1, x\alpha\sigma\theta\rho_2)$ and by induction hypothesis, $\sigma\theta\rho_1 \in [\![\exists x\ B]\!]_\phi$ and $\sigma\theta\rho_2 \in [\![A]\!]_\phi$. Hence, these proof-terms strongly terminate. Let n and n' be the maximum length of reduction sequences issued from these proof-terms. We prove by induction on $n + n'$ that $\delta_\exists(\sigma\theta\rho_1, x\alpha\sigma\theta\rho_2) \in [\![A]\!]_\phi$. Using Proposition 9.4, we only need to prove that every of its one step reducts is in $[\![A]\!]_\phi$. If the reduction takes place in $\sigma\theta\rho_1$ or $\sigma\theta\rho_2$ then we apply the induction hypothesis. Otherwise, if $\sigma\theta\rho_1$ has the form $\langle t, \rho'\rangle$ and the reduct is $(\rho'/\alpha)(t/x)\sigma\theta\rho_2 = ((\rho'/\alpha) \circ (t/x)\sigma)((t/x) \circ \theta)\rho_2$. By the definition of $[\![\exists x\ B]\!]_\phi$, there exists en element E in $\mathcal{M}_T$ (where T is the sort of x) such that $\rho' \in [\![B]\!]_{\phi,x=E}$. Thus, by induction hypothesis again, $((\rho'/\alpha) \circ (t/x)\sigma)((t/x) \circ \theta)\rho_2 \in [\![A]\!]_{\phi,x=E} = [\![A]\!]_\phi$.

Corollary 9.1 *Let $\mathcal{T}, \equiv$ be a theory in Deduction modulo theory, that has a model valued in the algebra $\mathcal{C}$. Then every proof-term in this theory strongly terminates.*

Proof. Let π be a proof-term of type A in a context Γ, θ be the substitution mapping each variable to itself, ϕ be a valuation mapping each variable of sort T to an element of $\mathcal{M}_T$, and σ be the substitution mapping each proof-term variable bound to proposition B in Γ to itself. Note that, by Proposition 9.2, this variable is an element of $[\![B]\!]_\phi$. Then $\pi = \sigma\theta\pi$ is an element of $[\![A]\!]_\phi$. Hence it strongly terminates.

Corollary 9.2 *Let $\mathcal{T}, \equiv$ be a super-consistent theory in Deduction modulo theory. Then, every proof-term in this theory strongly terminates.*

Proof. The algebra of candidates is full, ordered and complete.

Corollary 9.3 (Final rule) *Let $\varnothing, \equiv$ be a purely computational, super-consistent theory in Deduction modulo theory. If there exists a proof-term of type A in this theory, then there exists one that ends with an introduction rule.*

Proof. Consider a proof-term of type A in $\varnothing, \equiv$. This proof-term strongly terminates and its irreducible form is a closed and irreducible proof-term, by Proposition 8.4, it is an introduction.

Corollary 9.4 (Disjunction property, Witness property) *Let $\varnothing, \equiv$ be a purely computational, super-consistent theory in Deduction modulo theory. If there exists a proof-term of type $A \vee B$ this theory, then either there exists a proof-term of type A, or there exists a proof-term of type B in this theory.*

If there exists a proof-term of type $\exists x\ A$ in this theory, then there exists a term t and a proof-term of type $(t/x)A$ in this theory.

Proof. Consider a proof-term of type $A \vee B$, then, by Proposition 8.1, its irreducible form has has either the form $i(\pi_1)$ or $j(\pi_1)$. Consider a proof-term of type $\exists x\ A$, then, by Proposition 8.1, its irreducible form has the form $\langle t, \pi_1 \rangle$.

Corollary 9.5 (Gentzen's theorem) *Every proof-term in Arithmetic strongly terminates. Arithmetic has the disjunction and the witness property.*

Proof. By Proposition 4.10, arithmetic is super-consistent.

Corollary 9.6 (Girard's theorem) *Every proof-term in Simple type theory strongly terminates. Every proof-term in Simple type theory with Peano numbers strongly terminates. These theories have the disjunction and the witness property.*

Proof. By Proposition 6.4, Simple type theory is super-consistent. By Proposition 6.7, Simple type theory with Peano numbers is super-consistent.

9.4 Proof-term reduction in Arithmetic

Consider a class that contains zero and that is closed by the successor function. There are two ways to prove that this class contains the number 100. The first is to prove, by induction, that this class contains all the natural numbers, hence 100. The second

is to prove successively that it contains 1, 2, ..., 99, and then 100, using the closure by the successor function one hundred times. As we shall see, eliminating cuts in the first proof, in HA$^{\longrightarrow}$ or in Simple type theory with Peano numbers, yields the second.

Consider a class c and proof-terms π of type $0 \epsilon c$ and π' of type $\forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c)$. The proof-term $\lambda y\ \lambda\alpha\ (\alpha\ c\ \pi\ \pi')$ has the type $\forall y\ (N(y) \Rightarrow y \epsilon c)$, expressing that the class c contains all the natural numbers. Indeed, if α is a variable of type $N(y)$, that is equivalent to $\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c)$, the proof-term $(\alpha\ c\ \pi\ \pi')$ has type $y \epsilon c$. Thus, the proof-term corresponding to the first proof of the proposition $S^{100}(0) \epsilon c$ is

$$((\lambda y\ \lambda\alpha\ (\alpha\ c\ \pi\ \pi'))\ S^{100}(0)\ \rho_{100})$$

where ρ_{100} is a proof of $N(S^{100}(0))$. That corresponding to the second is

$$(\pi'\ S^{99}(0)\ \rho_{99}\ (\pi'\ S^{98}(0)\ \rho_{98}\ (...\ (\pi'\ 0\ \rho_0\ \pi))))$$

where ρ_0 is a proof-term of type $N(0)$, ρ_1 a proof-term of type $N(S(0))$, ρ_2 a proof-term of type $N(S^2(0))$, etc. Note that the proof-term π' occurs one hundred times in this proof-term.

There is only one irreducible proof-term of type $N(S^{100}(0))$, that is of $\forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow S^{100}(0) \epsilon c)$ and it is

$$\rho_{100} = \lambda c \lambda x \lambda f\ (f\ S^{99}(0)\ \rho_{99}\ (f\ S^{98}(0)\ \rho_{98}\ (...\ (f\ 0\ \rho_0\ x))))$$

Thus, the proof-term $((\lambda y\ \lambda\alpha\ (\alpha\ c\ \pi\ \pi'))\ S^{100}(0)\ \rho_{100})$ reduces to the proof-term $(\rho_{100}\ c\ \pi\ \pi')$ and then to $(\pi'\ S^{99}(0)\ \rho_{99}\ (\pi'\ S^{98}(0)\ \rho_{98}\ (...\ (\pi'\ 0\ \rho_0\ \pi))))$.

The key role in this reduction is played not by the term $S^{100}(0)$ but by the proof-term ρ_{100} of type $N(S^{100}(0))$. This proof-term is an iterator that copies the proof-term π' one hundred times.

We have seen in Chapter 6 that numbers could be defined as iterators: Church numbers. The proof-terms ρ_0, ρ_1, ρ_2, ... are variants of Church numbers, called *Parigot numbers*. They are defined by

$$\rho_0 = \lambda c \lambda x \lambda f\ x$$

$$\rho_{n+1} = \lambda c \lambda x \lambda f\ (f\ S^n(0)\ \rho_n\ (\rho_n\ c\ x\ f))$$

while Church numbers are defined by

$$\rho'_0 = \lambda c \lambda x \lambda f\ x$$

$$\rho'_{n+1} = \lambda c \lambda x \lambda f\ (f\ S^n(0)\ (\rho'_n\ c\ x\ f))$$

Note that Parigot numbers are proof-terms of type $N(S^n(0))$ modulo the rule

$$N(y) \longrightarrow \forall c\ (0 \epsilon c \Rightarrow \forall x\ (N(x) \Rightarrow x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c)$$

while Church numbers are proof-terms of type $N(S^n(0))$ modulo the rule

$$N(y) \longrightarrow \forall c\ (0 \epsilon c \Rightarrow \forall x\ (x \epsilon c \Rightarrow S(x) \epsilon c) \Rightarrow y \epsilon c)$$

9.5 Proofs as programs

In $HA^{\longrightarrow}$, or in Simple type theory with Peano numbers, there exists a proof-term π of type

$$\forall x\ (N(x) \Rightarrow \exists y\ (N(y) \wedge (x = 0 \Rightarrow y = 0) \wedge (\neg x = 0 \Rightarrow y = S(0))))$$

If we apply this proof-term to the term $S^n(0)$ and the proof-term ρ_n, we obtain the proof-term $(\pi\ S^n(0)\ \rho_n)$ of type

$$\exists y\ (N(y) \wedge (S^n(0) = 0 \Rightarrow y = 0) \wedge (\neg S^n(0) = 0 \Rightarrow y = S(0)))$$

The irreducible form of this proof-term has the form $\langle t, \langle \pi_1, \pi_2 \rangle\rangle$ where t is an irreducible term expressing a natural number, of the form $S^p(0)$, π_1 is a proof-term of type $N(t)$, and π_2 is a proof-term of type

$$(S^n(0) = 0 \Rightarrow t = 0) \wedge (\neg S^n(0) = 0 \Rightarrow t = S(0))$$

As this last proposition is provable, $p = \chi(n)$ where χ is the function that takes the value 0 at 0 and the value 1 at other natural numbers.

In this example, the function χ is specified by the proposition

$$(x = 0 \Rightarrow y = 0) \wedge (\neg x = 0 \Rightarrow y = S(0))$$

and the proof-term π is a program computing the function defined by this specification.

More generally, in arithmetic, any computable function f can be specified by a proposition A, such that the proposition $(n/x, p/y)A$ has a proof if and only if $p = f(n)$. When the proposition

$$\forall x\ (N(x) \Rightarrow \exists y\ (N(y) \wedge A))$$

has a proof π, the function f is said to be *provably total* in arithmetic. In this case, the irreducible form of the proof-term $(\pi\ S^n(0)\ \rho_n)$ has the form $\langle t, \langle \pi_1, \pi_2 \rangle\rangle$ and $t = S^p(0)$ for $p = f(n)$.

Thus, the language of proof-terms is a programming language, where all programs terminate and where unlike in Simply typed λ-calculus the function χ, and all functions that are provably total in arithmetic, can be expressed. The fact that more functions can be expressed in the language of proof-terms than in Simply typed λ-calculus comes from the fact that proof-terms are typed modulo a congruence, in particular modulo the reduction rule

$$N(y) \longrightarrow \forall c\ (0 \in c \Rightarrow \forall x\ (N(x) \Rightarrow x \in c \Rightarrow S(x) \in c) \Rightarrow y \in c)$$

that permits to type Parigot numbers, that are iterators that permit to define functions by induction on one of their arguments.

Exercise 9.1 *Leivant, Krivine, and Parigot have proposed to specify functions, not by a proposition, but by a congruence. Consider, for instance, the super-consistent (see Exercise 4.5) extension of $HA^{\longrightarrow}$ with the rules*

$$\chi(0) \longrightarrow 0$$

$$\chi(S(n)) \longrightarrow S(0)$$

1. *In this theory, give a proof-term π of type*

$$\forall x\ (N(x) \Rightarrow \exists y\ (N(y) \wedge y = \chi(x)))$$

2. *Consider a natural number n. What is the type of the proof-term $(\pi\ S^n(0)\ \rho_n)$? Show that its irreducible form has the form $\langle t, \langle \pi_1, \pi_2 \rangle \rangle$. What is t? What is π_1? What is π_2?*

3. *In this theory, give a proof-term π of type*

$$\forall x\ (N(x) \Rightarrow N(\chi(x)))$$

4. *Consider a natural number n. What is the type of the proof-term $(\pi\ S^n(0)\ \rho_n)$? What is its irreducible form π_1?*

Chapter 10

Dependent types

In Chapters 8 and 9, we have introduced a language to express proofs in Predicate logic and in Deduction modulo theory. This language is distinct from the logical language itself. For instance, if we represent proofs in arithmetic, the expression $S(0)$ is a *term* of arithmetic, the expression $S(0) = S(0) \Rightarrow S(0) = S(0)$ is a *proposition* of arithmetic and the expression $\lambda\alpha : (S(0) = S(0))\ \alpha$ is a *proof* of arithmetic. Moreover, the language of proofs contains two symbols λ—one corresponding to the $\Rightarrow$-intro rule and the other to the $\forall$-intro rule—two applications, two notions of pairs, etc.

This profusion of syntactic categories and symbols has the advantage of avoiding the confusion between distinct concepts, but it is also sometimes useful to have a smaller formalism. This is in particular the case when implementing such formalisms, as we may want, for instance, to avoid programming different substitution functions on terms, propositions, proofs, etc. The *λ-calculus with dependent types*, or $\lambda\Pi$-calculus, is an example of such a language allowing to express, terms, propositions, and proofs, in a uniform way.

10.1 The $\lambda\Pi$-calculus

The $\lambda\Pi$-calculus can be considered independently of the expression of terms, propositions, and proofs. Thus, we shall, in a first step, present the calculus for itself and in a second step explain how it can be used to express terms, propositions and proofs.

10.1.1 Dependent types

An example of *dependent type* is that of arrays of natural numbers. In most programming languages, the size of the array is part of its type. This way, there is not a single type $array$ but a type family $(array\ 0)$, $(array\ 1)$, $(array\ 2)$, etc. for the arrays of size 0, 1, 2, etc. Let us consider a function that maps a natural number n to the array of size n that contains only zeros

$$(f\ 0) = [\,]$$

$$(f\ 1) = [0]$$

$$(f\ 2) = [0, 0]$$

$$(f\ 3) = [0, 0, 0]$$

etc.

The argument of this function always has the type nat of natural numbers, but the type of the result of the function is not always the same. This type is $(array\ x)$ where x is the argument of the function. We could give the type $nat \to (array\ x)$ to this function, but this notation fails to express that the occurrence of x refers to the argument of the function. Thus, we must introduce another notation indicating, not only the type of the argument of the function, but also its name. We write this type $(x : nat) \to (array\ x)$ or $\Pi x : nat\ (array\ x)$.

The notation $A \to B$ becomes a particular case of the notation $\Pi x : A\ B$ when the variable x is not used in B and therefore needs not be indicated.

10.1.2 Types as terms

The language of the types of $\lambda\Pi$-calculus is more complex than that of Simply typed λ-calculus. Indeed the expressions $(array\ 0)$, $(array\ 1)$, $\Pi n : nat\ (array\ n)$ are types but the expressions $(array\ 0\ 0)$ or $(array\ true)$ are not. In the same way, the context $y : (array\ x)$ is not well-formed because the variable x is not declared anywhere.

Thus, type formation rules and context formation rules, similar to the term formation rules, are needed. We may consider that types are just particular terms and introduce a constant $Type$ to be the type of types.

We consider judgments of the form $\Gamma \vdash t : A$ that express that the term t has type A in the context Γ and also another form of judgments: Γ *well-formed* that express that Γ is a well-formed context. A particular case of the first form of judgment is the judgment $\Gamma \vdash t : Type$ that expresses that t is a type.

Before we express the rules of $\lambda\Pi$-calculus, let us give, once again, the definition of Simply typed λ-calculus in this framework.

Definition 10.1 (The Simply typed λ-calculus with types as terms)
The empty context is well-formed:

$$\frac{}{[\,]\ \textit{well-formed}}$$

Declaration of a type variable:

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma, A : Type\ \textit{well-formed}}$$

Formation of functional types:

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type}{\Gamma \vdash A \to B : Type}$$

Declaration of a variable:

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A \textit{ well-formed}}$$

Variables are terms:

$$\frac{\Gamma \textit{ well-formed}}{\Gamma \vdash x : A}\, x : A \in \Gamma$$

Abstraction:

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\, t : A \to B}$$

Application:

$$\frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\, t') : B}$$

10.1.3 Kinds

In the λΠ-calculus, for the terms $(array\ 0)$, $(array\ 1)$, etc. to have type $Type$, the variable $array$ must have the type $nat \to Type$. As the term $nat \to Type$ is the type of the variable $array$, we might want to give it the type $Type$. In the same way, as the term $Type$ is the type of the term $(array\ 0)$, we might want to give it the type $Type$. Unfortunately, if we give the type $Type$ to the constant $Type$, it is possible to type non terminating terms, and a theory based on such a calculus would be inconsistent, as shown by Girard's paradox. We thus have to introduce a new constant $Kind$ for the type of the terms $Type$, $nat \to Type$, ... Hence, there are four categories of terms

- $Kind$,
- kinds: $Type$, $nat \to Type$, ... whose type is $Kind$,
- types and type families: nat, $(array\ 0)$, $array$, ... whose type is a kind,
- objects: 0, $[0]$, ... whose type is a type.

Terms are formed as follows.

- $Kind$ is the only term in its category.
- Kinds are $Type$ and the kinds formed as a product (for instance, $nat \to Type$, that is $\Pi x : nat\ Type$).
- The types and type families are the variables of type and type families (for instance, nat, $array$, etc.) and the types and type families formed by application (for instance, $(array\ 0)$), abstraction (for instance, $\lambda n : nat\ (array\ (S\ n))$), and product (for instance, $\Pi n : nat\ (array\ n)$ and $nat \to nat$, that is $\Pi x : nat\ nat$).
- The objects are object variables (for instance, 0) and the objects formed by application (for instance, $(S\ 0)$) and abstraction (for instance, $\lambda x : nat\ x$).

10.1.4 The $\lambda\Pi$-calculus

Substitution is defined as usual avoiding variable capture, reduction is β-reduction, and equivalence is β-equivalence.

Proposition 10.1 (Confluence) *Reduction is confluent in the $\lambda\Pi$-calculus.*

The typing rules are the following.

Definition 10.2 (The $\lambda\Pi$-calculus)
The empty context is well-formed:

$$\frac{}{[\,]\ \textit{well-formed}}$$

Declaration of a type or type family variable:

$$\frac{\Gamma \vdash A : Kind}{\Gamma, x : A\ \textit{well-formed}}$$

Declaration of an object variable:

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A\ \textit{well-formed}}$$

$Type$ is a kind:

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma \vdash Type : Kind}$$

Variables are terms:

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma \vdash x : A}\ x : A \in \Gamma$$

Product (for kinds)

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}$$

Product (for types)

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}$$

Abstraction (for type families):

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}$$

Abstraction (for objects):

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}$$

Application:

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : (t'/x)B}$$

Conversion:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : Type \quad \Gamma \vdash B : Type}{\Gamma \vdash t : B} A \equiv B$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : Kind \quad \Gamma \vdash B : Kind}{\Gamma \vdash t : B} A \equiv B$$

The two final rules, called *Conversion* rules, permit to do a little bit of Deduction modulo theory in the λΠ-calculus. They are useful in the following case: consider the type family $array' = \lambda n : nat\ (array\ (S\ n))$, the type $(array'\ n)$ is the type of arrays of $n + 1$ elements. Applying the term $array'$ to 0, yields the term $(\lambda n : nat\ (array\ (S\ n))\ 0)$ that is convertible to $(array\ (S\ 0))$ but that *is not* the term $(array\ (S\ 0))$. This way, the conversion rule is needed for the term $[0]$, that has the type $(array\ (S\ 0))$, to have also type $(array'\ 0)$, that is $(\lambda n : nat\ (array\ (S\ n))\ 0)$.

The product rule for kinds allows to build the kind $nat \to Type$ but not $Type \to Type$. It is therefore possible to consider arrays parametrized by the number of their elements but not by the type of their elements. As we shall see in Chapter 12, other extensions of Simply typed λ-calculus (with polymorphic types, etc.) allow such constructions.

Proposition 10.2 (Uniqueness of type) *If $\Gamma \vdash t : A$ and $\Gamma \vdash t : A'$ are derivable, then $A \equiv A'$.*

Proof. By induction over derivation structure.

Proposition 10.3 (Kind) *If $\Gamma \vdash t : A$ is derivable, then $t \neq Kind$.*

Proof. By induction over derivation structure.

Proposition 10.4 (Products) *If $\Gamma \vdash \Pi x : A\ B : C$ is derivable, then $C \equiv Type$ or $C \equiv Kind$, $\Gamma \vdash A : Type$ is derivable and $\Gamma, x : A \vdash B : C$ is derivable.*

Proof. By induction over the structure of derivations. The last rule can only be a *Product* rule or a *Conversion* rule.

Proposition 10.5 (Abstractions) *If $\Gamma \vdash \lambda x : A\ u : C$ is derivable, then there exists a term B such that $\Gamma \vdash A : Type$ is derivable, $\Gamma, x : A \vdash B : Type$ or $\Gamma, x : A \vdash B : Kind$ is derivable, $\Gamma, x : A \vdash u : B$ is derivable and either $C = \Pi x : A\ B$ or $C \equiv \Pi x : A\ B$ and both C and $\Pi x : A\ B$ have type $Type$ or $Kind$ in Γ.*

Proof. By induction over the structure of derivations. The last rule can only be an *Abstraction* rule or a *Conversion* rule.

Proposition 10.6 (Applications) *If $\Gamma \vdash (u\ v) : A$ is derivable, then there exists B and C such that $\Gamma \vdash u : \Pi x : B\ C$ is derivable, $\Gamma \vdash v : B$ is derivable, and either $A = (v/x)C$ or $A \equiv (v/x)C$ and both A and $(v/x)C$ have type $Type$ or $Kind$ in Γ.*

Proof. By induction over the structure of derivations. The last rule can only be an *Application* rule or a *Conversion* rule.

Proposition 10.7 (Weakenning) *If* $\Delta\Delta'$ well-formed *is derivable, and* $\Delta \vdash B : Type$ *or* $\Delta \vdash B : Kind$ *is derivable, then* $\Delta, x : B, \Delta'$ well-formed *also is.*

If $\Delta\Delta' \vdash t : A$ *is derivable, and* $\Delta \vdash B : Type$ *or* $\Delta \vdash B : Kind$ *is derivable, then* $\Delta, x : B, \Delta' \vdash t : A$ *also is.*

Proof. By induction over derivation structure.

Proposition 10.8 (Substitution) *If* $\Delta, x : A, \Delta'$ well-formed *and* $\Delta \vdash u : A$ *are derivable, then* $\Delta, (u/x)\Delta'$ well-formed *also is.*

If $\Delta, x : A, \Delta' \vdash t : B$ *and* $\Delta \vdash u : A$ *are derivable, then* $\Delta, (u/x)\Delta' \vdash (u/x)t : (u/x)B$ *also is.*

Proof. By induction over derivation structure.

Proposition 10.9 (Subject reduction) *If* $\Gamma \vdash t : A$ *is derivable and* $t \longrightarrow^* t'$ *then* $\Gamma \vdash t' : A$ *is derivable,*

Proof. We first prove that if $\Gamma \vdash t : A$ is derivable and t reduces in one step at the root to t', then $\Gamma \vdash t' : A$ is derivable. As t reduces in one step at the root to t', then $t = ((\lambda x : B\ u)\ v)$ and $t' = (v/x)u$. By Proposition 10.6, as $\Gamma \vdash ((\lambda x : B\ u)\ v) : A$ is derivable, there exists B' and C' such that $\Gamma \vdash \lambda x : B\ u : \Pi x : B'\ C'$ is derivable, $\Gamma \vdash v : B'$ is derivable, and either $A = (v/x)C'$ or $A \equiv (v/x)C'$ and both terms have type $Type$ or $Kind$ in Γ. By Proposition 10.5, as $\Gamma \vdash \lambda x : B\ u : \Pi x : B'\ C'$ is derivable, there exists a C such that $\Gamma \vdash B : Type$ is derivable, $\Gamma, x : B \vdash C : Type$ or $\Gamma, x : B \vdash C : Kind$ is derivable, $\Gamma, x : B \vdash u : C$ is derivable, and either $\Pi x : B'\ C' = \Pi x : B\ C$ or $\Pi x : B'\ C' \equiv \Pi x : B\ C$ and both terms have type $Type$ or $Kind$ in Γ. Using Proposition 10.1, either $B = B'$ or $B \equiv B'$ and both terms have type $Type$ in Γ and either $C = C'$ or $C \equiv C'$, C has type $Type$ or $Kind$ in $\Gamma, x : B$ and C' has this same type in $\Gamma, x : B'$. As $\Gamma \vdash v : B'$ is derivable, $\Gamma \vdash v : B$ is derivable, either because $B = B'$ or because $B \equiv B'$ and both terms have type $Type$ in Γ.

Using Proposition 10.8, $\Gamma \vdash (v/x)u : (v/x)C$ is derivable, that is $\Gamma \vdash t' : (v/x)C$ is derivable. Using Proposition 10.8 again, either $(v/x)C = (v/x)C'$, or $(v/x)C \equiv (v/x)C'$ and both terms have type $Type$ or $Kind$ in Γ. Then, $\Gamma \vdash t' : (v/x)C'$ is derivable, either because $(v/x)C = (v/x)C'$, or because $(v/x)C \equiv (v/x)C'$ and both terms have type $Type$ or $Kind$ in Γ. Finally, $\Gamma \vdash t' : A$ is derivable, either because $A = (v/x)C'$ or because $A \equiv (v/x)C'$ and both terms have type $Type$ or $Kind$ in Γ.

Then, we conclude that if $\Gamma \vdash t : A$ is derivable and $t \longrightarrow^1 t'$ then $\Gamma \vdash t' : A$ is derivable, by induction over the structure of t and that if $\Gamma \vdash t : A$ is derivable and $t \longrightarrow^* t'$ then $\Gamma \vdash t' : A$ by induction over the structure of reduction sequences.

Proposition 10.10 (Contexts) *If* $\Delta, x : A, \Delta'$ well-formed *is derivable or if* $\Delta, x : A, \Delta' \vdash t : B$ *is derivable then* $\Delta \vdash A : Type$ *or* $\Delta \vdash A : Kind$ *is derivable.*

Proof. By induction over derivation structure.

Proposition 10.11 (Types are well-typed) *If* $\Gamma \vdash t : A$ *is derivable, then either* $A = Kind$*, or there exists a* B *such that* $\Gamma \vdash A : B$ *is derivable.*

Proof. By induction on the structure of the derivation of $\Gamma \vdash t : A$, using Propositions 10.10 and 10.7 for the case of the rule *Variable* and the Propositions 10.8 and 10.4 for the case of the rule *Application*.

Definition 10.3 (Objects, type families, kinds) *Let t be a term such that $\Gamma \vdash t : A$. The term t*

- *is an* object *in Γ, if $\Gamma \vdash A : Type$,*
- *is a* type family *in Γ, if $\Gamma \vdash A : Kind$,*
- *is a* kind *in Γ, if $A = Kind$.*

Note that, as $Type$ has type $Kind$, types are type families.

Theorem 10.1 (Classification) *If $\Gamma \vdash t : A$ is derivable, then the term t is either an object, a type family or a kind in Γ, and these cases are exclusive.*

Proof. By induction on the structure of the derivation of $\Gamma \vdash t : A$.

- The last rule cannot be *Empty* or a *Declaration* rule.
- If the last rule is the *Type* rule, then $A = Kind$ and t is a kind in Γ.
- If it is the *Variable* rule, then t is a variable x and $x : A$ is an element of Γ, hence, by Propositions 10.10 and 10.7, $\Gamma \vdash A : Type$ or $\Gamma \vdash A : Kind$ is derivable, thus t is an object or a type family in Γ.
- If it is a *Product* rule, then $A = Type$ or $A = Kind$, thus t is a type family or a kind in Γ.
- If it is an *Abstraction* rule, then $A = \Pi x : B\ C$ and, by Proposition 10.4, the type of A is either $Type$ or $Kind$, thus t is either an object or a type family in Γ.
- If it is the *Application* rule, then $t = (u\ v)$, u has type $\Pi x : B\ C$ in Γ, v has type B in Γ, and $A = (u/x)C$, by Proposition 10.4, C has type $Type$ or $Kind$ in $\Gamma, x : B$, thus, by Proposition 10.8, $(v/x)C$ has type $(v/x)Type = Type$ or $(v/x)Kind = Kind$ and u is an object or a type family in Γ.
- If it is the *Conversion* rule, then A has type $Type$ or $Kind$ and t is an object or a type family in Γ.

To prove that a term cannot be be both a kind and a type family, a kind and an object, a type family and an object, we first note that if $\Gamma \vdash t : A$ is derivable, then $t \not\equiv Kind$. Assume $t \equiv Kind$. By Proposition 10.1, there exists a term t' such that $t \longrightarrow^* t'$ and $Kind \longrightarrow^* t'$. As $Kind$ is irreducible, $t' = Kind$, and, by Proposition 10.9, $\Gamma \vdash Kind : A$. A contradiction with Proposition 10.3.

Then, if t is both a kind and a type family, $A \equiv Kind$, and $\Gamma \vdash A : Kind$, a contradiction.

If t is both a kind and an object, then $A \equiv Kind$ and $\Gamma \vdash A : Type$, a contradiction.

If t is both a type family and an object, then $\Gamma \vdash A : Kind$, $\Gamma \vdash A : B$, and $\Gamma \vdash B : Kind$. Thus, by Proposition 10.2, $Kind \equiv B$, a contradiction.

10.2 The termination of reduction in the $\lambda\Pi$-calculus

To prove the termination of reduction in the $\lambda\Pi$-calculus, we first need to adapt the notion of candidate to the new language of terms: candidates in the $\lambda\Pi$-calculus are sets of terms of the $\lambda\Pi$-calculus.

Definition 10.4 (Candidates in the $\lambda\Pi$-calculus) *Let C be a set of terms and S be a set of sets of terms. The set $\tilde{\Pi}(C, S)$ is defined as the set of strongly terminating terms t such that if $t \longrightarrow^* \lambda x : E\ t_1$ then for all t' in C, and for all D in S, $(t'/x)t_1 \in D$. Let C and D be two sets of terms. The set $C \Rightarrow D$ is defined as $\tilde{\Pi}(C, \{D\})$.*

Candidates are inductively defined by the three rules

- *the set of all strongly terminating terms in a candidate,*
- *if C is a candidate and S is a set of candidates, then $\tilde{\Pi}(C, S)$ is a candidate.*
- *if S is a set of candidates, then $\bigcap S$ is a candidate.*

We write $\mathcal{C}$ for the set of all candidates.

Proposition 10.12 (Termination) *If C is a candidate, then all the elements of C strongly terminate.*

Proof. By induction on the construction of C.

Proposition 10.13 (Variables) *Let C be a candidate and x be a variable, then $x \in C$.*

Proof. By induction on the construction of C. The variable x strongly terminates and it never reduces to an abstraction.

Proposition 10.14 (Closure by reduction) *If C is a candidate, t is an element of C, and $t \longrightarrow^* t'$, then t' is an element of C.*

Proof. By induction on the construction of C. As t is an element of C it strongly terminates, thus t' strongly terminates. Then, if $C = \tilde{\Pi}(D, S)$, and $t' \longrightarrow^* \lambda x : A\ t_1$, then $t \longrightarrow^* \lambda x : A\ t_1$, and for all u in D, and for all E in S, $(u/x)t_1 \in E$. Thus $t' \in C$.

Proposition 10.15 (Applications) *Let C be a candidate. If all the one-step reducts of the term $(u_1\ u_2)$ are in C, then $(u_1\ u_2)$ is in C.*

Proof. By induction on the construction of C. We first prove that $(u_1\ u_2)$ strongly terminates. Let $(u_1\ u_2) = t_1, t_2, \ldots$ be a reduction sequence issued from $(u_1\ u_2)$. If this sequence is empty it is finite. Otherwise, we have $(u_1\ u_2) \longrightarrow^1 t_2$ and hence t_2 is an element of C thus it strongly terminates and the reduction sequence is finite.

Then, if $C = \tilde{\Pi}(D, S)$ and $(u_1\ u_2) \longrightarrow^* \lambda x : A\ v$, then let $(u_1\ u_2) = t_1, t_2, \ldots, t_n = \lambda x : A\ v$ be a reduction sequence from $(u_1\ u_2)$ to $\lambda x : A\ v$. As $(u_1\ u_2)$ is an application and $\lambda x : A\ v$ is not, $n \geq 2$. Thus $(u_1\ u_2) \longrightarrow^1 t_2 \longrightarrow^* \lambda x : A\ v$. We have $t_2 \in C$ and $t_2 \longrightarrow^* \lambda x : A\ C$, thus for all w in D and E in S, $(w/x)v \in E$.

Thus $(u_1\ u_2) \in C$.

Definition 10.5 *We define a family of set $(\mathcal{M}_t)_t$ indexed by terms of the λΠ-calculus as follows.*

- $\mathcal{M}_{Type} = \mathcal{M}_{Kind} = \mathcal{C}$,
- $\mathcal{M}_x = \{e\}$, *an arbitrary singleton,*
- $\mathcal{M}_{\lambda x:A\ t} = \mathcal{M}_t$,
- $\mathcal{M}_{(t\ u)} = \mathcal{M}_t$,
- $\mathcal{M}_{\Pi x:A\ B}$ *is the set of functions f from $\mathcal{M}_A$ to $\mathcal{M}_B$, except if $\mathcal{M}_B = \{e\}$, in which case $\mathcal{M}_{\Pi x:A\ B} = \{e\}$, or if $\mathcal{M}_A = \{e\}$, in which case $\mathcal{M}_{\Pi x:A\ B} = \mathcal{M}_B$.*

Note that, if t is an object or a type family, then $\mathcal{M}_t = \{e\}$ and if t is a kind or $t = Kind$, then $\mathcal{M}_t = \mathcal{C}$. Thus, this definition could be greatly simplified: if t is an object or a type family, then $\mathcal{M}_t = \{e\}$ and if t is a kind or $t = Kind$, then $\mathcal{M}_t = \mathcal{C}$. However we shall keep the Definition above that generalizes better.

Proposition 10.16 *If t and u are two well-typed tems in a context Γ and $t \equiv u$ then $\mathcal{M}_t = \mathcal{M}_u$.*

Proof. First note that if u is an object, then $\mathcal{M}_{(u/x)t} = \mathcal{M}_t$. Thus $\mathcal{M}_{((\lambda x:A\ t)\ u)} = \mathcal{M}_t = \mathcal{M}_{(u/x)t}$. We conclude using confluence and subject reduction.

Definition 10.6 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a well-formed context. A* Γ-valuation *ϕ is a function mapping every variable x_i to an element of $\mathcal{M}_{A_i}$.*

Definition 10.7 *Let t be a term of λΠ-calculus, that has type A in a context Γ or that is equal to $Kind$. Let ϕ be a Γ-valuation. The element $[\![t]\!]_\phi$ of $\mathcal{M}_A$ is defined as follows.*

- $[\![Type]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![Kind]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![x]\!]_\phi = \phi(x)$,
- $[\![\lambda x : C\ t]\!]_\phi$ *is the function of domain $\mathcal{M}_C$ mapping a in $\mathcal{M}_C$ to $[\![t]\!]_{\phi,x=a}$, except if $[\![t]\!]_{\phi,x=a} = e$ for all a, in which case $[\![\lambda x : C\ t]\!]_\phi = e$, or if $\mathcal{M}_C = \{e\}$, in which case $[\![\lambda x : C\ t]\!]_\phi = [\![t]\!]_{\phi,x=e}$,*
- $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi\ [\![u]\!]_\phi$, *except if $[\![t]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = e$, or if $[\![u]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi$,*
- $[\![\Pi x : C\ D]\!]_\phi$ *is the candidate* $\tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$.

Note that, as $[\![t]\!]_\phi$ is an element of $\mathcal{M}_A$, if t is an object, then $[\![t]\!]_\phi = e$ and if t is a type family, a kind or the symbol $Kind$, then $[\![t]\!]_\phi$ is a candidate. Thus, this Definition could be simplified by defining the function $[\![.]\!]$ as a function mapping type families, kinds and the symbol $Kind$ to candidates. Note also that, in an abstraction $\lambda x : C\ t$, the term C always has type $Type$, thus $\mathcal{M}_C$ is always equal to $\{e\}$ and that in an application $(t\ u)$, u is always an object, thus $[\![u]\!]_\phi$ is always equal to $\{e\}$. Thus, this Definition could be further simplified:

- $[\![Type]\!]_\phi$ is the set of strongly terminating terms,
- $[\![Kind]\!]_\phi$ is the set of strongly terminating terms,
- $[\![x]\!]_\phi = \phi(x)$,
- $[\![\lambda x : C\ t]\!]_\phi = [\![t]\!]_{\phi, x=e}$,
- $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi$,
- $[\![\Pi x : C\ D]\!]_\phi$ is the candidate $\tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$.

However we shall keep the Definition above that generalizes better.

Proposition 10.17 *If t and u are well-typed in some context Γ and $t \equiv u$, then $[\![t]\!]_\phi = [\![u]\!]_\phi$.*

Proof. Consider a well-typed term of the form $((\lambda x : B\ t)\ u)$. As this term is well-typed, u has type B and thus $[\![u]\!]_\phi \in \mathcal{M}_B$. Let C be the type of t, if $\mathcal{M}_C \neq \{e\}$ then $[\![((\lambda x : B\ t)\ u)]\!]_\phi = [\![t]\!]_{\phi, x=[\![u]\!]_\phi} = [\![(u/x)t]\!]_\phi$. Otherwise, $[\![((\lambda x : B\ t)\ u)]\!]_\phi = e = [\![(u/x)t]\!]_\phi$. We conclude using confluence and subject reduction.

Proposition 10.18 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a context, ϕ be a Γ-valuation, σ be a substitution mapping every x_i to an element of $[\![A_i]\!]_\phi$ and t a term of type B in Γ. Then $\sigma t \in [\![B]\!]_\phi$.*

Proof. By induction on the structure of t.

- If $t = Type$, then $B = Kind$, $[\![B]\!]_\phi$ is the set of all strongly terminating terms and $\sigma t = Type \in [\![B]\!]_\phi$.
- If $t = x$ is a variable, then by definition of σ, $\sigma t \in [\![A]\!]_\phi$.
- If $t = \Pi x : C\ D$, then $B = Type$ or $B = Kind$, $\Gamma \vdash C : Type$ and $\Gamma, x : C \vdash D : Type$ or $\Gamma, x : C \vdash D : Kind$, by induction hypothesis $\sigma C \in [\![Type]\!]_\phi$, that is σC strongly terminates and $\sigma D \in [\![Type]\!]_\phi$ or $\sigma D \in [\![Kind]\!]_\phi$, that is σD strongly terminates. Thus $\sigma(\Pi x : C\ D) = \Pi x : \sigma C\ \sigma D$ strongly terminates also and it is an element of $[\![B]\!]_\phi$.
- If $t = \lambda x : C\ u$ where u has type D. Then $B = \Pi x : C\ D$ and $[\![B]\!]_\phi$ is the set of terms s such that s strongly terminates and if s reduces to $\lambda x : E\ s_1$ then for all s' in $[\![C]\!]_\phi$ and all a in $\mathcal{M}_C$, $(s'/x)s_1$ is an element of $[\![D]\!]_{\phi,x=a}$.

  We have $\sigma t = \lambda x : \sigma C\ \sigma u$, consider a reduction sequence issued from this term. This sequence can only reduce the terms σC and σu. By induction hypothesis, the term σC is an element of $[\![Type]\!]_\phi$ and the term σu is an element of $[\![D]\!]_\phi$, thus the reduction sequence is finite.

  Furthermore, every reduct of σt has the form $\lambda x : C'\ v$ where C' is a reduct of σC and v is a reduct of σu. Let w be any term of $[\![C]\!]_\phi$, and a be any element of $\mathcal{M}_C$, the term $(w/x)v$ can be obtained by reduction from $((w/x) \circ \sigma)u$. By induction hypothesis, the term $((w/x) \circ \sigma)u$ is an element of $[\![D]\!]_{\phi,x=a}$. Hence, by Proposition 10.14 the term $(w/x)v$ is an element of $[\![D]\!]_{\phi,x=a}$. Therefore, the term $\sigma\lambda x\ u$ is an element of $[\![B]\!]_\phi$.

- If the term t has the form $(u_1\ u_2)$ then u_1 is a term of type $\Pi x : C\ D$, u_2 a term of type C and $B = (u_2/x)D$.

  We have $\sigma t = (\sigma u_1\ \sigma u_2)$, and by induction hypothesis $\sigma u_1 \in [\![\Pi x : C\ D]\!]_\phi$ and $\sigma u_2 \in [\![C]\!]_\phi$. Hence these terms are strongly terminating. Let n be the maximum length of a reduction sequence issued from σu_1 and n' the maximum length of a reduction sequence issued from σu_2. We prove by induction on $n + n'$ that $(\sigma u_1\ \sigma u_2) \in [\![B]\!]_\phi$. Using Proposition 10.15, we only need to prove that every of its one step reducts is in $[\![B]\!]_\phi$. If the reduction takes place in σu_1 or in σu_2 then we apply the induction hypothesis. Otherwise, σu_1 has the form $\lambda x : E\ u'$ and the reduct is $(\sigma u_2/x)u'$. By the definition of $[\![\Pi x : C\ D]\!]_\phi$ this term is in $[\![D]\!]_{\phi, x=[\![u_2]\!]_\phi} = [\![(u_2/x)D]\!]_\phi = [\![B]\!]_\phi$.

Corollary 10.1 *Let Γ be a context and t be a term well-typed in Γ. Then t strongly terminates.*

Proof. Let B be the type of t in Γ, let ϕ be any Γ-valuation, σ be the substitution mapping every x_i to itself. Note that, by Proposition 10.13, this variable is an element of $[\![A_i]\!]_\phi$. Then $t = \sigma t \in [\![B]\!]_\phi$. Hence it strongly terminates.

10.3 Representation of terms, propositions, and proofs of minimal logic

We consider a fragment of Predicate logic where the only connective is the implication $\Rightarrow$ and the only quantifier is the universal quantifier $\forall$. Thus the only deduction rules are the axiom rule and the introduction and elimination rules of the implication and of the universal quantifier. This fragment is called *minimal logic*.

Terms, propositions and proofs of minimal logic can be expressed in the $\lambda\Pi$-calculus.

Definition 10.8 (Languages as contexts) *To each language $\mathcal{L}$ of Predicate logic, we associate a context Ξ containing*

- *for each sort s of $\mathcal{L}$ a variable s of type $Type$,*
- *for each function symbol f of arity $\langle s_1, ..., s_n, s' \rangle$, a variable, also written f, of type $s_1 \to ... \to s_n \to s'$,*
- *for each predicate symbol P of arity $\langle s_1, ..., s_n \rangle$ a variable, also written P, of type $s_1 \to ... \to s_n \to Type$.*

Definition 10.9 (Embedding of terms and propositions) *Let t be a term (resp. A a proposition) of a language $\mathcal{L}$, we embed the term t (resp. the proposition A) as a term of the $\lambda\Pi$-calculus as follows:*

- $\Phi(x) = x$,
- $\Phi(f(t_1, ..., t_n)) = (f\ \Phi(t_1)\ ...\ \Phi(t_n))$,

- $\Phi(P(t_1, ..., t_n)) = (P\ \Phi(t_1)\ ...\ \Phi(t_n))$,
- $\Phi(A \Rightarrow B) = \Phi(A) \to \Phi(B)$, *that is* $\Pi x : \Phi(A)\ \Phi(B)$,
- $\Phi(\forall x\ A) = \Pi x : s\ \Phi(A)$.

Proposition 10.19 *Let t be a term (resp. A a proposition) of a language $\mathcal{L}$, let Γ be a context containing all the variables of Ξ and for each variable x of sort s free in t (resp. A), a variable, also written x, of type s. Then $\Gamma \vdash \Phi(t) : s$ (resp. $\Gamma \vdash \Phi(A) : Type$).*

Proof. By induction over the structure of t (resp. A).

Proposition 10.20 (Embedding of proofs) *Let $\mathcal{L}$ be a language and $A_1, ..., A_n \vdash B$ be a sequent of $\mathcal{L}$. Consider a context Γ containing the variables of Ξ, for each variable x of sort s free in $A_1, ..., A_n \vdash B$, a variable, also written x, of type s and for each hypothesis A_i a variable α_i of type $\Phi(A_i)$. Then, if the sequent $A_1, ..., A_n \vdash B$ has a proof in Natural deduction, there exists a term π in $\lambda\Pi$-calculus such that $\Gamma \vdash \pi : \Phi(B)$.*

Proof. By induction over the structure of the proof of $A_1, ..., A_n \vdash B$.

Remark. To express the proofs of Predicate logic, the *Product (for kinds)* rule is needed, for instance to build the type $s \to Type$, that is the type of unary predicate symbols. But the *Abstraction (for type families)* rule that allows to build the term $array' = \lambda x : nat\ (array\ (S\ x))$ and the *Conversion* rules that permit to give both types $(array'\ 0)$ and $(array\ (S\ 0))$ to the term $[0]$, are not needed. Indeed, in Predicate logic, there is no way to build a predicate, only the predicates given as predicate symbols can be used.

10.4 The $\lambda 1$-calculus

We now want to express proofs not only of minimal logic, but also of full Predicate logic. This leads to extend the $\lambda\Pi$-calculus to the $\lambda 1$-calculus. Our presentation of the $\lambda 1$-calculus will be shorter, as many aspects of the $\lambda 1$-calculus still remain to be investigated.

10.4.1 Sums, disjoint unions, unit and empty types

In the same way as the type $A \to B$ can be generalized to a type $\Pi x : A\ B$ allowing to express the dependency of the target type of a function with respect to the value of its argument, the Cartesian product type $A \times B$ can be generalized to a type $\Sigma x : A\ B$ allowing to express the dependency of the type of the second component of the ordered pair with respect to the first. This way, the ordered pairs $\langle 0, [\ ]\rangle$, $\langle 1, [0]\rangle$, $\langle 2, [0, 0]\rangle$, $\langle 3, [0, 0, 0]\rangle$, ... can all be all given the type $\Sigma x : nat\ (array\ x)$. Note that the ordered pair $\langle 3, [0, 0, 0]\rangle$ has both types $\Sigma x : nat\ (array\ x)$ and $\Sigma x : nat\ (array\ 3)$. In order to avoid losing the uniqueness of typing we indicate the type of each pair as an extra argument of the symbol $\langle , \rangle$: $\langle t, u\rangle_{\Sigma x:A\ B}$. Finally, if t is a pair, we write *fst*(t) for its first component and *snd*(t) for its second.

We write $A + B$ for the disjoint union of A and B. We write $(i^{A+B}\ t)$ for the element of the disjoint union corresponding to t in the first type, and $(j^{A+B}\ u)$ for the element of the disjoint union corresponding to the element u in the second type. We write $\delta(t, x : A\ u, y : B\ v)$ for the object defined by cases on t.

We write $\top$ for the singleton type and I for its element.

We write $\bot$ for the empty type and $\delta^A_\bot(t)$ an object of type A built from an object in $\bot$.

10.4.2 The $\lambda 1$-calculus

Definition 10.10 ($\lambda 1$-calculus) *The typing rules of $\lambda 1$-calculus are those of $\lambda\Pi$-calculus, and the following rules.*

$$\frac{}{\Gamma \vdash \top : Type}$$

$$\frac{}{\Gamma \vdash \bot : Type}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Sigma x : A\ B : Type}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type}{\Gamma \vdash A + B : Type}$$

$$\frac{}{\Gamma \vdash I : \top}$$

$$\frac{\Gamma \vdash t : \bot}{\Gamma \vdash \delta^A_\bot(t) : A}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma \vdash t : A \quad \Gamma \vdash u : (t/x)B}{\Gamma \vdash \langle t, u\rangle_{\Sigma x:A\ B} : \Sigma x : A\ B}$$

$$\frac{\Gamma \vdash t : \Sigma x : A\ B}{\Gamma \vdash \mathit{fst}(t) : A}$$

$$\frac{\Gamma \vdash t : \Sigma x : A\ B}{\Gamma \vdash \mathit{snd}(t) : (\mathit{fst}(t)/x)B}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad \Gamma \vdash t : A}{\Gamma \vdash i^{A+B}(t) : A + B}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad \Gamma \vdash u : B}{\Gamma \vdash j^{A+B}(u) : A + B}$$

$$\frac{\Gamma \vdash t : A + B \quad \Gamma \vdash C : Type \quad \Gamma, x : A \vdash u : C \quad \Gamma, y : B \vdash v : C}{\Gamma \vdash \delta(t, x : A\ u, y : B\ v) : C}$$

Substitution is defined as usual, avoiding variable capture, and reduction with the following reduction rules.

Definition 10.11 (Reduction)

$$((\lambda x : A\ t)\ u) \longrightarrow (u/x)t$$
$$\mathit{fst}(\langle t, u\rangle_{\Sigma x:A\ B}) \longrightarrow t$$
$$\mathit{snd}(\langle t, u\rangle_{\Sigma x:A\ B}) \longrightarrow u$$
$$\delta(i^{A+B}(t), x : A\ u, y : B\ v) \longrightarrow (t/x)u$$
$$\delta(j^{A+B}(t), x : A\ u, y : B\ v) \longrightarrow (t/y)v$$

Proposition 10.21 (Uniqueness of type) *If* $\Gamma \vdash t : A$ *and* $\Gamma \vdash t : A'$*, then* $A \equiv A'$.

10.4.3 Representation of terms, propositions, and proofs

Consider a language $\mathcal{L}$ in Predicate logic. We consider a context Ξ as in Definition 10.8, we embed terms as in Definition 10.9, but we extend the embedding of Propositions as follows.

- $\Phi(P(t_1, ..., t_n)) = P(\Phi(t_1), ..., \Phi(t_n))$,
- $\Phi(\top) = \top$,
- $\Phi(\bot) = \bot$,
- $\Phi(A \wedge B) = \Phi(A) \times \Phi(B) = \Sigma x : \Phi(A)\ \Phi(B)$,
- $\Phi(A \vee B) = \Phi(A) + \Phi(B)$,
- $\Phi(A \Rightarrow B) = \Phi(A) \rightarrow \Phi(B)$, that is $\Pi x : \Phi(A)\ \Phi(B)$,
- $\Phi(\forall x\ A) = \Pi x : s\ \Phi(A)$,
- $\Phi(\exists x\ A) = \Sigma x : s\ \Phi(A)$.

Then Proposition 10.20 generalizes to full Predicate logic.

10.5 The $\lambda\Pi$-calculus modulo theory

10.5.1 Proofs in Deduction modulo theory

To represent the proofs of Deduction modulo theory, we extend the $\lambda\Pi$-calculus to the $\lambda\Pi$-calculus modulo theory. First we need to define the notion of reduction rule in this framework. A *reduction rule* is a triple $l \longrightarrow^{\Gamma} r$ where Γ is a context and l and r are β-irreducible terms. If $l \longrightarrow^{\Gamma} r$ is a reduction rule θ is a substitution binding the variables of Γ, then we say that the term θl *reduces* to the term θr.

If $\mathcal{R}$ is a reduction system, that is a set of reduction rules, then the *congruence generated by* $\mathcal{R}$, $\equiv_{\mathcal{R}}$, is the smallest congruence such that if t reduces to u, then $t \equiv_{\mathcal{R}} u$ and the *congruence generated by* β *and* $\mathcal{R}$, $\equiv_{\beta\mathcal{R}}$, is the smallest congruence such that if t β-reduces to u or t reduces to u then $t \equiv_{\mathcal{R}} u$.

A rewrite system is said to be *well-typed* if for all contexts Δ and for all terms t, u and A, if $\Delta \vdash t : A$ and t reduces to u, then $\Delta \vdash u : A$.

Definition 10.12 (λΠ-modulo theory) *Let Ξ be a context and $\mathcal{R}$ a reduction system in Ξ. Let $\equiv_{\beta\mathcal{R}}$ be the congruence of terms generated by the rules of $\mathcal{R}$ and the rule β.*

The λΠ-calculus modulo *$\mathcal{R}$ is the extension of the λΠ-calculus obtained by replacing the relation $\equiv_\beta$ by $\equiv_{\beta\mathcal{R}}$ in the conversion rules*

$$\frac{\Gamma \vdash A : Type \;\; \Gamma \vdash B : Type \;\; \Gamma \vdash t : A}{\Gamma \vdash t : B} \; A \equiv_{\beta\mathcal{R}} B$$

$$\frac{\Gamma \vdash A : Kind \;\; \Gamma \vdash B : Kind \;\; \Gamma \vdash t : A}{\Gamma \vdash t : B} \; A \equiv_{\beta\mathcal{R}} B$$

Extending the congruence in this way jeopardizes confluence—hence subject reduction—and termination: confluence and termination do not hold for all congruences, but, as we shall see, for some.

Proofs in minimal Deduction modulo theory can be expressed as terms in the λΠ-calculus modulo theory in the same way as proofs in minimal logic can be expressed as terms in λΠ-calculus.

10.5.2 The expression of the proofs of minimal Simple type theory

Minimal Simple type theory is the restriction of Simple type theory, where the symbols $\dot{\top}$, $\dot{\bot}$, $\dot{\wedge}$, $\dot{\vee}$, and $\dot{\exists}_A$ and the associated reduction rules are dropped. Minimal simple type theory can be expressed in minimal Deduction modulo theory. Hence its proofs can be expressed in λΠ-calculus modulo theory.

But, as Simple type theory contains an infinite number of sorts and symbols we have to adapt the translation. Instead of introducing a variable of type $Type$ for each sort, we introduce two variables ι and o of type $Type$ and translate the simple type as follows

- $|\iota| = \iota$, $|o| = o$,
- $|A \to B| = |A| \to |B|$, that is $\Pi x : |A| \; |B|$.

In the same way, to translate terms, we use, in Simple type theory, a notation for terms based on λ-calculus and not on combinators and we translate terms as follows

- $|x| = x$,
- $|(t\ u)| = (|t| \; |u|)$,
- $|(\lambda x : A \; t)| = \lambda x : |A| \; |t|$.

The idea here is that λΠ-calculus already contains a notion of function that may be reused instead of redefining one for Simple type theory. Thus, the β-reduction rules of Simple type theory is simply simulated by the β-reduction rule of λΠ-calculus.

But, we keep the symbol ε, $\dot{\Rightarrow}$ and $\dot{\forall}_A$ and the rules

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow \varepsilon(x) \to \varepsilon(y)$$

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \Pi y : |A| \; \varepsilon(x\ y)$$

Note that we still have an infinite number of symbols, as we have a symbol $\dot{\forall}_A$ for each simple type A. But, in any proof, only a finite number of such symbols are used. Thus we consider a context $\Xi = [\iota : Type, o : Type, \varepsilon : o \to Type, \dot{\Rightarrow} : o \to o \to o, \dot{\forall}_A : (A \to o) \to o]$ for a finite number of simple types A. It is possible to make A a parameter of $\dot{\forall}$ but this requires to introduce contents for simple types, that is variable T of type $Type$, $\dot{\iota}$ and $\dot{o}$ or type T, $\dot{\to}$ of type $T \to T \to T$, and η of type $T \to Type$, and rules

$$\eta(\dot{\iota}) \longrightarrow \iota$$

$$\eta(\dot{o}) \longrightarrow o$$

$$\eta(x \dot{\to} y) \longrightarrow \eta(x) \to \eta(y)$$

but we shall not discuss this system here.

We want to prove the termination of $\lambda\Pi$-calculus modulo these rules. We first prove the termination of β-reduction.

Definition 10.13 *We define a family of set $(\mathcal{M}_t)_t$ indexed by terms of the $\lambda\Pi$-calculus modulo theory as follows.*

- $\mathcal{M}_{Type} = \mathcal{M}_{Kind} = \mathcal{M}_o = \mathcal{C}$,
- $\mathcal{M}_\iota = \mathcal{M}_\varepsilon = \mathcal{M}_x = \mathcal{M}_{\dot{\Rightarrow}} = \mathcal{M}_{\dot{\forall}_A} = \{e\}$, *an arbitrary singleton,*
- $\mathcal{M}_{\lambda x:A\ t} = \mathcal{M}_t$,
- $\mathcal{M}_{(t\ u)} = \mathcal{M}_t$,
- $\mathcal{M}_{\Pi x:A\ B}$ *is the set of functions f from $\mathcal{M}_A$ to $\mathcal{M}_B$, except if $\mathcal{M}_B = \{e\}$, in which case $\mathcal{M}_{\Pi x:A\ B} = \{e\}$, or if $\mathcal{M}_A = \{e\}$ in which case $\mathcal{M}_{\Pi x:A\ B} = \mathcal{M}_B$.*

Note that if t is an object, then we have $\mathcal{M}_t = \{e\}$ but this is not the case anymore for types as $\mathcal{M}_o = \mathcal{C}$.

Proposition 10.22 *If t and u are two well-typed tems in a context Γ and $t \equiv u$ then $\mathcal{M}_t = \mathcal{M}_u$.*

Proof. First note that if the term u is an object, then $\mathcal{M}_{(u/x)t} = \mathcal{M}_t$. Thus $\mathcal{M}_{((\lambda x:A\ t)\ u)} = \mathcal{M}_t = \mathcal{M}_{(u/x)t}$.

Then, as for all v, $\mathcal{M}_{(\varepsilon\ v)} = \mathcal{M}_\varepsilon = \{e\}$, and if $\mathcal{M}_D = \{e\}$, then $\mathcal{M}_{\Pi x:C\ D} = \{e\}$, then

$$\mathcal{M}_{(\varepsilon\ C)\to(\varepsilon\ D)} = \{e\} = \mathcal{M}_{(\varepsilon\ (C \dot{\Rightarrow} D))}$$

and

$$\mathcal{M}_{\Pi x:|C|(\varepsilon\ (D\ x))} = \{e\} = \mathcal{M}_{(\varepsilon\ (\dot{\forall}_C\ D))}$$

We conclude using confluence and subject reduction.

Definition 10.14 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a well-formed context. A Γ-valuation ϕ is a function mapping every variable x_i to an element of $\mathcal{M}_{A_i}$.*

Definition 10.15 *Let t be a term of λΠ-calculus, that has type A in a context Γ or that is equal to $Kind$. Let ϕ be a Γ-valuation. The element $[\![t]\!]_\phi$ of $\mathcal{M}_A$ is defined as follows.*

- $[\![Type]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![Kind]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![o]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![\iota]\!]_\phi$ *is the set of strongly terminating terms,*
- $[\![x]\!]_\phi = \phi(x)$*,*
- $[\![\lambda x : C\ t]\!]_\phi$ *is the function of domain $\mathcal{M}_C$ mapping a in $\mathcal{M}_C$ to $[\![t]\!]_{\phi,x=a}$, except if $[\![t]\!]_{\phi,x=a} = e$ for all a, in which case $[\![\lambda x : C\ t]\!]_\phi = e$, or if $\mathcal{M}_C = \{e\}$, in which case $[\![\lambda x : C\ t]\!]_\phi = [\![t]\!]_{\phi,x=e}$,*
- $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi([\![u]\!]_\phi)$*, except if $[\![t]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = e$, or if $[\![u]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi$,*
- $[\![\Pi x : C\ D]\!]_\phi$ *is the candidate $\tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$,*
- $[\![\varepsilon]\!]_\phi$ *is the identity on $\mathcal{C}$,*
- $[\![\dot{\Rightarrow}]\!]_\phi = \tilde{\Rightarrow}$*,*
- $[\![\dot{\forall}_A]\!]_\phi$ *is the function mapping the function f from $\mathcal{M}_{|A|}$ to $\mathcal{C}$ to the candidate $\tilde{\Pi}([\![|A|]\!]_\phi, \{f(a) \mid a \in \mathcal{M}_{|A|}\})$.*

As $[\![t]\!]_\phi$ is an element of $\mathcal{M}_A$, if t has type $Type$ or $Kind$ then $[\![t]\!]_\phi$ is a candidate. If $t = Kind$ then $[\![t]\!]_\phi$ is a candidate also.

Proposition 10.23 *If t and u are well-typed in a context Γ and $t \equiv u$ then $[\![t]\!]_\phi = [\![u]\!]_\phi$.*

Proof. Consider a well-typed term of the form $((\lambda x : B\ t)\ u)$. As this term is well-typed, u has type B and thus $[\![u]\!]_\phi \in \mathcal{M}_B$. Let C be the type of t, if $\mathcal{M}_C \neq \{e\}$ then $[\![((\lambda x : B\ t)\ u)]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi} = [\![(u/x)t]\!]_\phi$. Otherwise, $[\![((\lambda x : B\ t)\ u)]\!]_\phi = e = [\![(u/x)t]\!]_\phi$.

Then we have $[\![\varepsilon(\dot{\Rightarrow}\ t\ u)]\!]_\phi = ([\![t]\!]_\phi \tilde{\Rightarrow} [\![u]\!]_\phi) = [\![\varepsilon(t) \to \varepsilon(u)]\!]_\phi$ and $[\![\varepsilon(\dot{\forall}_A\ t)]\!]_\phi = \tilde{\Pi}([\![|A|]\!]_\phi, \{[\![t]\!]_\phi(a) \mid a \in \mathcal{M}_{|A|}\}) = [\![\Pi y : |A|\ \varepsilon(t\ y)]\!]_\phi$.

We conclude using confluence and subject reduction.

Proposition 10.24 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a context, ϕ be a Γ-valuation, σ be a substitution mapping every x_i to an element of $[\![A_i]\!]_\phi$ and t a term of type B in Γ. Then $\sigma t \in [\![B]\!]_\phi$. Hence t strongly terminates for β-reduction.*

Proof. Same as that of Theorem 10.18 and Corollary 10.1.

Corollary 10.2 *All well-typed terms strongly terminate for $\beta\mathcal{R}$-reduction.*

Proof. Proceed like in the proof of Proposition 7.10.

Remark. In the proof of termination of reduction for Simple type theory (Corollary 9.6), we used first the super-consistency of Simple type theory (Proposition 6.4), where the set $\mathcal{M}_{A\to B}$ is defined as the set of functions from $\mathcal{M}_A$ to $\mathcal{M}_B$. Then, we used the fact that the super-consistency of Simple type theory implies the existence of a model valued in the algebra $\mathcal{C}$ (Corollary 9.2). In this model, the proposition $A \Rightarrow B$ is interpreted as the candidate $[\![A]\!]_\phi \tilde{\Rightarrow} [\![B]\!]_\phi$ where $\tilde{\Rightarrow}$ is defined (Definition 9.1) as the function $\tilde{\Rightarrow}$ mapping the set of terms E and F to the set of terms π such that π strongly terminates and when it reduces to a term of the form $\lambda\alpha : A\ \pi_1$ then for every π' in E, $(\pi'/\alpha)\pi_1 \in F$.

The proof above follows the same lines, except that both sorts and propositions of Simple type theory are terms of the $\lambda\Pi$-calculus modulo theory. Thus, the set of indices of the family $\mathcal{M}$ and the domain of the function $[\![.]\!]$ are the set of the terms of the $\lambda\Pi$-calculus modulo theory.

10.5.3 The $\lambda 1$-calculus modulo theory

Definition 10.16 ($\lambda 1$-modulo theory) *Let Ξ be a context and $\mathcal{R}$ a reduction system in Ξ. Let $\equiv_{\beta\mathcal{R}}$ be the congruence of terms generated by the rules of $\mathcal{R}$ and the rule β.*

The $\lambda 1$-calculus modulo $\mathcal{R}$ *is the extension of the $\lambda 1$-calculus obtained by replacing the relation $\equiv_\beta$ by $\equiv_{\beta\mathcal{R}}$ in the conversion rules*

$$\frac{\Gamma \vdash A : Type\ \ \Gamma \vdash B : Type\ \ \Gamma \vdash t : A}{\Gamma \vdash t : B}\ A \equiv_{\beta\mathcal{R}} B$$

$$\frac{\Gamma \vdash A : Kind\ \ \Gamma \vdash B : Kind\ \ \Gamma \vdash t : A}{\Gamma \vdash t : B}\ A \equiv_{\beta\mathcal{R}} B$$

Proofs of Deduction modulo theory can be expressed as terms of the $\lambda 1$-calculus modulo theory in the same way as proofs of minimal deduction modulo theory can be expressed as terms of the $\lambda\Pi$-calculus modulo theory.

Proofs of Simple type theory can be expressed as terms of the $\lambda 1$-calculus modulo theory in the same way as proofs of minimal Simple type theory can be expressed in the $\lambda\Pi$-calculus modulo theory.

The axiom of choice

$$\forall x : \iota\ \exists y : \iota\ \varepsilon(P\ x\ y) \Rightarrow \exists f : \iota \to \iota\ \forall x : \iota\ \varepsilon(P\ x\ (f\ x))$$

is not provable in Simple type theory, but it is in the $\lambda 1$-calculus.

Proposition 10.25 (The theorem of choice) *The type*

$$\Pi x : \iota\ \Sigma y : \iota\ \varepsilon(P\ x\ y) \Rightarrow \Sigma f : \iota \to \iota\ \Pi x : \iota\ \varepsilon(P\ x\ (f\ x))$$

is inhabited in $\lambda 1$-calculus.

Proof. It is inhabited by the term

$$t = \lambda u : H\ \langle \lambda x : \iota\ \mathit{fst}(u\ x), \lambda x : \iota\ \mathit{snd}(u\ x)\rangle$$

where H is the type $\Pi x : \iota\ \Sigma y : \iota\ \varepsilon(P\ x\ y)$.

This difference is explained by the fact that as terms of the logic and proof-terms are mixed, we can build terms using proofs, for example the term $\lambda x : \iota\ \mathit{fst}(u\ x)$ that uses the proof u of $\forall x\ \exists y\ \varepsilon(P\ x\ y)$. In particular the projections *fst* and *snd* are very different from the $\exists$-elim rule.

Chapter 11

Inductive types

In Chapter 4, we have seen that Arithmetic could be expressed as an axiomatic theory with or without Peano's predicate symbol N, but that this symbol was needed to express Arithmetic in Deduction modulo theory as a purely computational theory in which proof reduction terminates. Indeed, Arithmetic without the predicate symbol N has the disjunction property for closed propositions, but not for open ones: the proposition $x = 0 \vee \exists y\ (x = S(y))$ can be proved by induction on x, but neither $x = 0$ nor $\exists y\ (x = S(y))$ are provable. As already noted, in the formulation of arithmetic with Peano's predicate symbol symbol N, this proposition is expressed as $N(x) \Rightarrow (x = 0 \vee \exists y\ (x = S(y)))$, that is not a disjunction.

11.1 Arithmetic without Peano's symbol

When Arithmetic is defined without Peano's symbol, we can still transform most of the axioms into reduction rules,

$$y \in f_{x_1,...,x_n,y,A}(x_1, ..., x_n) \longrightarrow A$$

$$x = y \longrightarrow \forall c\ (x \in c \Rightarrow y \in c)$$

$$\mathit{Pred}(0) \longrightarrow 0$$

$$\mathit{Pred}(S(x)) \longrightarrow x$$

$$0 + y \longrightarrow y$$

$$S(x) + y \longrightarrow S(x + y)$$

$$0 \times y \longrightarrow 0$$

$$S(x) \times y \longrightarrow x \times y + y$$

$$\mathit{Null}(0) \longrightarrow \top$$

$$\mathit{Null}(S(x)) \longrightarrow \bot$$

but induction remains an axiom

$$\forall c\ (0 \in c \Rightarrow \forall x\ (x \in c \Rightarrow S(x) \in c) \Rightarrow \forall y\ y \in c)$$

This axiom can be replaced by a deduction rule

$$\frac{\Gamma \vdash 0 \in c \quad \Gamma \vdash \forall x\ (x \in c \Rightarrow S(x) \in c)}{\Gamma \vdash t \in c}$$

Thus, when we express the proofs as terms, we need to introduce a symbol for this axiom or this rule. For instance if c is a class, π a proof of the proposition $0 \in c$, π' a proof of the proposition $\forall x\ (x \in c \Rightarrow S(x) \in c)$, and t a term, we can write $Rec(c, \pi, \pi', t)$ the proof of the proposition $t \in c$ build with this deduction rule.

This jeopardizes the property that cut-free proofs end with an introduction. For instance, let $c = f_{y,(y=0 \vee \exists z\ (y=S(z)))}$ be the class of numbers that are either zero or a successor. The term $\pi = i(\sigma)$, where σ is a proof of $0 = 0$, is a proof of the proposition $0 = 0 \vee \exists z\ (0 = S(z))$ that is $0 \in c$. The term $\pi' = \lambda x\ \lambda\alpha\ (j(\langle x, \sigma' \rangle))$, where σ' is a proof of $S(x) = S(x)$, is a proof of the proposition $\forall x\ (x \in c \Rightarrow S(x) \in c)$ and the term $Rec(c, \pi, \pi', S(S(0)))$ a proof of the proposition $S(S(0)) \in c$, that is $S(S(0)) = 0 \vee \exists z\ (S(S(0)) = S(z))$. This term is thus an irreducible proof of a disjunction and it does not end with an introduction rule of the disjunction.

In order to recover the property that *closed* cut-free proofs end with an introduction rule, we extend the notion of cut in arithmetic and mimic the behavior of proofs by induction in the formulation of arithmetic with Peano's predicate symbol. To do so, we add two rules allowing to reduce a proof built with the induction rule and a specific natural number n to proof obtained by iterating n times the proof of the proposition $\forall x\ (x \in c \Rightarrow S(x) \in c)$ (see Chapter 9)

$$Rec(c, \pi, \pi', 0) \longrightarrow \pi$$

$$Rec(c, \pi, \pi', S(x)) \longrightarrow (\pi'\ x\ Rec(c, \pi, \pi', x))$$

This way the proof $Rec(c, \pi, \pi', S(S(0)))$ reduces to $j(\langle S(0), (S(0)/x)\sigma' \rangle)$ that ends with an introduction rule of the disjunction.

As proofs are functions, we can consider, in a first step, the extension of simply typed λ-calculus with such a recursion operator independently of the representation of proofs. This leads to an extension of Simply typed λ-calculus, called Gödel System T.

11.2 Gödel System T

Definition 11.1 (Gödel System T) *Gödel System T is an extension of Simply typed λ-calculus with a single base type nat and constants $0 : nat$, $S : nat \to nat$, and for each type A, a symbol Rec^A of arity $\langle A, nat \to A \to A, nat, A \rangle$. The reduction rules are*

$$((\lambda x : A\ t)\ u) \longrightarrow (u/x)t$$

$$Rec^A(a, g, 0) \longrightarrow a$$

$$Rec^A(a, g, (S\ n)) \longrightarrow (g\ n\ Rec^A(a, g, n))$$

Example 11.1 *The function that multiplies its argument by two is defined by the term*

$$d = \lambda a : nat\ Rec^{nat}(0, \lambda x : nat\ \lambda y : nat\ (S\ (S\ y)), a)$$

The addition is defined by the term

$$+ = \lambda a : nat\ \lambda b : nat\ Rec^{nat}(a, \lambda x : nat\ \lambda y : nat\ (S\ y), b)$$

The multiplication is defined by the term

$$\times = \lambda a : nat\ \lambda b : nat\ Rec^{nat}(0, \lambda x : nat\ \lambda y : nat\ (+\ y\ a), b)$$

The power function is defined by the term

$$\uparrow = \lambda a : nat\ \lambda b : nat\ Rec^{nat}((S\ 0), \lambda x : nat\ \lambda y : nat\ (\times\ y\ a), b)$$

The predecessor is defined by the term

$$pred = \lambda a : nat\ Rec^{nat}(0, \lambda x : nat\ \lambda y : nat\ x, a)$$

The characteristic function of the singleton $\{0\}$ *is defined by the term*

$$\chi_{\{0\}} = \lambda a : nat\ Rec^{nat}((S\ 0), \lambda x : nat\ \lambda y : nat\ 0, a)$$

The characteristic function of the set of non-zero numbers is defined by the term

$$\chi_{\mathbb{N}\setminus\{0\}} = \lambda a : nat\ Rec^{nat}(0, \lambda x : nat\ \lambda y : nat\ (S\ 0), a)$$

The characteristic function of the set of even numbers is defined by the term

$$\chi_{2\mathbb{N}} = \lambda a : nat\ Rec^{nat}((S\ 0), \lambda x : nat\ \lambda y : nat\ (\chi_{\{0\}}\ y), a)$$

Remark. A definition by induction of a many argument function is often expressed as follows

$$f(x_1, ..., x_{p-1}, 0) = a(x_1, ..., x_{p-1})$$
$$f(x_1, ..., x_{p-1}, S(n)) = g(x_1, ..., x_{p-1}, n, f(x_1, ..., x_{p-1}, n))$$

Such a function can always be expressed in the System T:

$$f = \lambda x_1 ... \lambda x_{p-1} \lambda x_p\ Rec^{nat}((a\ x_1\ ...\ x_{p-1}), \lambda n \lambda m\ (g\ x_1\ ...\ x_{p-1}\ n\ m), x_p)$$

Therefore, all primitive recursive functions can be expressed in the System T. But it is also possible to express non primitive recursive functions, such as Ackermann's function A defined by

$$A(0, x) = 2^x$$
$$A(S(n), 0) = 1$$
$$A(S(n), S(x)) = A(n, A(S(n), x))$$

$$A = \lambda n\ Rec^{nat \to nat}(P, \lambda p \lambda f \lambda m\ (Rec^{nat}\ (S\ 0)\ (\lambda q \lambda s\ (f\ s))\ m), n)$$

where $P : nat \to nat$ is a term of System T, expressing the function $x \mapsto 2^x$.

11.3 The termination of Gödel System T

To prove that the System T strongly terminates, we simulate the recursor of System T with Parigot numbers. We first define the following theory in Deduction modulo theory.

Definition 11.2 (The theory $\mathcal{T}$) *In Deduction modulo theory, consider a language with a single sort, a unary predicate symbol ε, a constant nat and a binary function symbol $\dot{\Rightarrow}$, and the theory defined by the reduction rules*

$$\varepsilon(nat) \longrightarrow \forall p\ (\varepsilon(p) \Rightarrow (\varepsilon(nat) \Rightarrow \varepsilon(p) \Rightarrow \varepsilon(p)) \Rightarrow \varepsilon(p))$$

$$\varepsilon(x \dot{\Rightarrow} y) \longrightarrow \varepsilon(x) \Rightarrow \varepsilon(y)$$

Proposition 11.1 *The theory $\mathcal{T}$ is super-consistent.*

Proof. Consider a full ordered and complete pre-Heyting algebra $\mathcal{B}$, for each element C of $\mathcal{B}$, consider the model $\mathcal{M}_C$ of domain $\mathcal{B}$ where the symbol ε is interpreted as the identity over $\mathcal{B}$, $\dot{\Rightarrow}$ is interpreted as $\tilde{\Rightarrow}$ and nat is interpreted as C. The function Φ mapping each element C of $\mathcal{B}$ to

$$\Phi(C) = [\![\forall p\ (\varepsilon(p) \Rightarrow (\varepsilon(nat) \Rightarrow \varepsilon(p) \Rightarrow \varepsilon(p)) \Rightarrow \varepsilon(p))]\!]^{\mathcal{M}_C}$$

is monotone because the occurrence of $\varepsilon(nat)$ in $\forall p\ (\varepsilon(p) \Rightarrow (\varepsilon(nat) \Rightarrow \varepsilon(p) \Rightarrow \varepsilon(p)) \Rightarrow \varepsilon(p))$ is positive. Hence it has a fixed-point C_0.

We build a model of domain $\mathcal{B}$ where the symbol ε is interpreted as the identity over $\mathcal{B}$, $\dot{\Rightarrow}$ is interpreted as $\tilde{\Rightarrow}$, and nat is interpreted as C_0. This model is a model of the theory $\mathcal{T}$.

Proposition 11.2 *Every term of the System T strongly terminates.*

Proof. The theory $\mathcal{T}$ is super-consistent, hence its proofs strongly terminate. We prove that strong termination for proofs of $\mathcal{T}$ implies strong termination for terms of System T. Types of the System T are terms of the theory $\mathcal{T}$ and terms of type A in the System T can be translated into proofs of $\varepsilon(A)$ as follows

- $|x| = x$, $|u\ v| = |u|\ |v|$, $|\lambda x : A\ u| = \lambda x\ |u|$,
- $|0| = \lambda p \lambda x \lambda f\ x$,
- $|S| = \lambda n \lambda p \lambda x \lambda f\ (f\ n\ (n\ p\ x\ f))$,
- $|Rec^A(t, u, n)| = (|n|\ A\ |t|\ |u|)$.

It is routine to check that if $t \longrightarrow^1 u$ in the System T then $|t| \longrightarrow^+ |u|$ in the theory $\mathcal{T}$:

$$|Rec^A(x\ f\ 0)| \longrightarrow^* (|0|\ A\ x\ f) = (\lambda p \lambda x \lambda f\ x)\ A\ x\ f \longrightarrow^+ x$$

$$|Rec^A(x, f, (S\ n))| \longrightarrow^* (|(S\ n)|\ A\ x\ f) = (\lambda p \lambda x \lambda f\ (f\ |n|\ (|n|\ p\ x\ f)))\ A\ x\ f)$$

$$\longrightarrow^+ (f\ |n|\ (|n|\ A\ x\ f)) = (f\ |n|\ |Rec^A(x, f, n)|) = |(f\ n\ Rec^A(x, f, n))|$$

11.4 Martin-Löf Type theory

In Deduction modulo theory, reduction rules apply to the terms and propositions of the theory, but the proof-reduction rules are restricted to the rules eliminating cuts, such as the β-reduction for implication cuts. In the $\lambda\Pi$-calculus and in the $\lambda 1$-calculus, the terms and the proofs are identified, thus nothing prevents to add reduction rules on proofs as well.

In particular, it is possible to express arithmetic without Peano's predicate symbol N, taking induction as a deduction rule and adding the reduction rules of System T to reduce cuts associated to this rule. This leads to Martin-Löf Type theory.

In the $\lambda 1$-calculus, we can express the first axiom of equality

$$\mathit{refl} : \forall x : nat\ (x = x)$$

We rather replace it by a deduction rule: *the reflexivity rule*

$$\frac{}{\Gamma \vdash t = t}$$

and to interpret this rule, we introduce a symbol *refl* such that for all t of type nat, *refl*(t) is a proof of $t = t$.

To express the second axiom of equality

$$\forall c \forall x \forall y\ (x = y \Rightarrow x\ \epsilon\ c \Rightarrow y\ \epsilon\ c)$$

we would need to introduce a sort κ for classes and a comprehension scheme. Instead, we prefer to use the fact that the $\lambda 1$-calculus already contains a notion of function that may be reused and give the type $nat \rightarrow Type$ to classes, and write $(P\ t)$ the proposition formerly written $t\ \epsilon\ P$ Yet, the axiom

$$\forall P : nat \rightarrow Type\ \forall x : nat\ \forall y : nat\ (x = y \Rightarrow (P\ x) \Rightarrow (P\ y))$$

cannot be expressed in the $\lambda 1$-calculus, because it is impossible to quantify over variables of type $nat \rightarrow Type$. But for each term P of type $nat \rightarrow Type$, we can take an axiom

$$\forall x : nat\ \forall y : nat\ (x = y \Rightarrow (P\ x) \Rightarrow (P\ y))$$

Again, we prefer to introduce a deduction rule: *the substitutivity rule*, also called *Leibniz' rule*,

$$\frac{\Gamma \vdash x = y \quad \Gamma \vdash (P\ x)}{\Gamma \vdash (P\ y)}$$

and to interpret this rule, we introduce a symbol L such that for all terms P of type $nat \rightarrow Type$, for all terms t and u of type nat, for all terms π of type $t = u$ and for all terms π' of type $(P\ t)$, the term $L(P, t, u, \pi, \pi')$ has type $(P\ u)$.

We take a symbol P_4 to interpret the axiom

$$\forall x : nat\ (0 = (S\ x) \Rightarrow \bot)$$

and a symbol Rec such that for all terms P of type $nat \to Type$, for all terms π of type of $(P\ 0)$, for all terms π' of type $\forall n : nat\ ((P\ n) \Rightarrow (P\ (S\ n)))$, and for all terms t of type nat, $Rec(P, \pi, \pi', t)$ has type $(P\ t)$.

Let P be a term of type $nat \to Type$, a a term of type nat and π a term of type $(P\ a)$.The term *refl*(a) is a proof of $a = a$ and the term $L(P, a, a,$ *refl*$(a), \pi)$ is another proof of $(P\ a)$.

This proof can be considered as a cut, as starting from a proof π of $(P\ a)$, we replace a by itself, the proof of $a = a$ being given by the reflexivity rule. Usual cuts are sequences formed with an introduction rule and an elimination rule on the same symbol. Here, the reflexivity rule can be considered as an introduction rule for equality and the substitutivity rule as an elimination rule for this symbol. Eliminating this cut leads to replace the proof $L(P, a, a,$ *refl*$(a), \pi)$ by the term π. Thus we add the rule

$$L(P, a, a, \textit{refl}(a), \pi) \longrightarrow \pi$$

In the same way, the symbols 0 and S can be considered as introduction rules for the symbol nat and the induction axioms as elimination rules and eliminating these cuts leads, as we have seen, to the rules

$$Rec(P, a, g, 0) \longrightarrow a$$

$$Rec(P, a, g, (S\ n)) \longrightarrow (g\ n\ Rec(P, a, g, n))$$

Definition 11.3 (Martin-Löf type theory) *Martin-Löf type theory is the extension of $\lambda 1$-calculus with the symbols $=$, refl, L, nat, 0, S, Rec, and P_4, and the reduction rules*

$$L(P, a, a, \textit{refl}(a), \pi) \longrightarrow \pi$$

$$Rec(P, a, g, 0) \longrightarrow a$$

$$Rec(P, a, g, (S\ n)) \longrightarrow (g\ n\ Rec(P, a, g, n))$$

Note that the symbol Rec can be used to define functions by induction and also to prove propositions by induction. Thus, the predecessor functions, the addition and the multiplication can be defined in Martin-Löf Type theory and we do not need to take them as primitive symbols like in Definition 4.7. In contrast, we cannot define by induction the predicate symbol $Null$ whose type $nat \to Type$ is a kind and thus we need to keep the axiom

$$\forall x : nat\ (0 = (S\ x) \Rightarrow \bot)$$

Proposition 11.3 *All terms in Martin-Löf type theory strongly terminate.*

Proposition 11.4 (Final rule) *Let t be an irreducible closed term in Martin-Löf Type theory. Then*

- *the term t does not have the type $\bot$,*
- *if the term t has type $\Sigma x : A\ B$ then it has the form $\langle v, w\rangle$,*
- *if the term t has type $A + B$ then it has the form $i(v)$ or $j(w)$,*

- *if the term t has type $v = w$ then it has the form refl(v), hence the terms v and w are identical,*
- *if the term t has type nat then it has the form 0 or $S(v)$.*

Proof. By induction on the structure of t. The term t has the form $(u\ c_1\ ...\ c_n)$ where u is not an application. We consider the following cases:

- u is $Type$ or a product, in this case, $n = 0$ and the type of t is a sort, contradicting the hypotheses,
- u has the form $\lambda x : A\ v$, in which case $n = 0$ and the type of t is a product, contradicting the hypotheses,
- u is I, in which case $n = 0$, and the term t has type $\top$, contradicting the hypotheses,
- u has the form $\langle v, w\rangle$, in which case $n = 0$,
- u has the form *fst*(v) or *snd*(v), in which case by induction hypothesis v is a pair, contradicting the fact that the term t is irreducible,
- u has the form $i(v)$ or $j(w)$, in which case $n = 0$,
- u has the form $\delta(v, f, g)$, in which case, v is either $i(v)$ or $j(w)$, contradicting the fact that the term t is irreducible,
- u has the form $\delta_\perp(v)$, in which case, v is a term of type $\perp$, and there is none,
- u is the constant nat, in which case $n = 0$, the type of t is $Type$, contradicting the hypotheses,
- u is the constant $=$ in which case the type of t is a product or $Type$ contradicting the hypotheses,
- u has the form *refl*(v), in which case $n = 0$,
- u has the form $L(P, t, u, \pi)$ in which case, by induction hypothesis π has the form *refl*(u), contradicting the fact that the term t is irreducible,
- u is the constant $0 : nat$,
- u is the constant $S : nat \to nat$, in which case $n = 1$,
- u has the form $Rec(P, a, g, t)$, in which case, by induction hypothesis $t = 0$ or $t = S(v)$, contradicting the fact that the term is irreducible,
- u is the constant P_4, in which case $n = 2$, $t = (P_4\ t\ u)$, u is a proof of the proposition $0 = (S\ t)$, and by induction these two terms are identical, a contradiction.

Corollary 11.1 (Disjunction, Witness) *If a proposition of the form $A \vee B$ has a* closed *proof, then either A is provable or B is. If a proposition of the form $\exists x : A\ B$ has a* closed *proof, then there exists a term t such that the proposition $(t/x)B$ is provable.*

11.5 Inductive types

Besides natural numbers, other datatypes can be defined in $\lambda 1$-calculus: lists, trees, etc.

For instance, we can define lists of natural numbers, with two constructors

$$nil : list$$

$$cons : nat \to list \to list$$

and a rule

$$\frac{a : (P\ nil) \quad g : \forall a : nat\, \forall l : list\, ((P\ l) \to (P\ (cons\ a\ l))) \quad l : list}{Rec(P, a, g, l) : (P\ l)}$$

that permits to build functions by induction on the structure of lists, and also to prove properties of list by induction.

Chapter 12

Polymorphism

In the previous chapter, we have shown how to express the terms, the propositions and the proofs of Simple type theory in $\lambda\Pi$-calculus modulo theory. The chosen formulation of Simple type theory uses a predicate symbol ε and thus a distinction between propositional contents—terms of type o—and propositions. As we have seen, we may also want to get rid of the predicate symbol ε and identify propositional contents with propositions.

Dropping the predicate symbol ε and identifying the term $(\dot{\Rightarrow}\ x\ y)$ with $x \to y$ and $\dot{\forall}_A\ x$ with $\Pi y : |A|\ (x\ y)$ permits to drop the reduction rules

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow \varepsilon(x) \to \varepsilon(y)$$

$$\varepsilon(\dot{\forall}_A\ x) \longrightarrow \Pi y : |A|\ \varepsilon(x\ y)$$

that would then reduce terms to identical terms.

Yet, identifying the symbols o and $Type$ is difficult because these two symbols have different types: o has type $Type$, but $Type$ has type $Kind$, and we cannot assign the type $Type$ to the symbol $Type$, as the rule $Type : Type$ leads to a system where some well-typed terms do not terminate and where all types are inhabited, hence all propositions provable.

This problem is an illustration of a tension between Simple type theory where there is a type o of propositional contents, that becomes the type of propositions when propositional contents are identified with propositions and the Curry-de Bruijn-Howard correspondence that leads to identify propositions and types. The first leads to $o : Type$ and the second to $o = Type$ and these two principles are incompatible.

One solution out of this dilemma is to keep the distinction between propositional contents and propositions, as in the Chapter 9. In this case, only some terms of type $Type$ have a content. In particular, o does not. Another is to abandon the idea that the symbol o has type $Type$ and assign it the type $Kind$. But then, to be able to build the type $o \to o$ and $(A \to o) \to o$, that is $Type \to Type$ and $(A \to Type) \to Type$, we have to extend the typing rules.

In the $\lambda\Pi$-calculus, we had the product rules

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}$$

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}$$

where the term B could have type $Type$ or $Kind$, but where the type A always had type $Type$. We now extend these rules, in order to allow the term A to have type $Kind$ as well

$$\frac{\Gamma \vdash A : Kind \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma, x : A \vdash B : Kind}{\Gamma \vdash \Pi x : A\ B : Kind}$$

and we extend the abstraction rule in a similar way. We get this way another λ-calculus where the terms, the propositions, and the proofs of Simple type theory can be expressed, that contains less reduction rules, but more typing rules: *The Calculus of constructions*.

These two new product rules allow to build type families parametrized not only by a term, but also by a type. For instance it is possible to parametrize the type of arrays not only by its size, but also by the type of its elements. The second new rule allows to build the kind $Type \to Type$, we can declare a variable $array$ of type $Type \to Type$ and build the types $(array\ nat)$, $(array\ bool)$, $(array\ (array\ nat))$, etc. This rule is called the rule of *type constructors*.

The first new rule in contrast permits to build the type $\Pi x : Type\ (x \to (array\ x) \to (array\ x))$ and declare a variable $cons$ of this type. This rule is called the rule of *polymorphic types*.

12.1 The Calculus of constructions

Definition 12.1 (The Calculus of constructions)
The empty context is well-formed:

$$\frac{}{[\,]\ \textit{well-formed}}$$

Declaration of a variable:

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A\ \textit{well-formed}}$$

where $s = Type$ or $s = Kind$.
$Type$ is a kind:

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma \vdash Type : Kind}$$

Variables are terms:

$$\frac{\Gamma \textit{ well-formed} \quad x : A \in \Gamma}{\Gamma \vdash x : A}$$

Product:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A\ B : s_2}$$

where $s_1 = Type$ *or* $s_1 = Kind$ *and* $s_2 = Type$ *or* $s_2 = Kind$*.*
Abstraction:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}$$

where $s_1 = Type$ *or* $s_1 = Kind$ *and* $s_2 = Type$ *or* $s_2 = Kind$*.*
Application:

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : (t'/x)B}$$

Conversion:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : s \quad \Gamma \vdash B : s \quad A \equiv B}{\Gamma \vdash t : B}$$

where $s = Type$ *or* $s = Kind$*.*

The termination of the Calculus of constructions can be proved in a similar way as the termination of $\lambda\Pi$-calculus modulo the two rules that allowed the expression of Simple Type Theory.

Definition 12.2 *We define a family of set* $(\mathcal{M}_t)_t$ *indexed by terms of the Calculus of Constructions as follows.*

- $\mathcal{M}_{Type} = \mathcal{M}_{Kind} = \mathcal{C}$*,*
- $\mathcal{M}_x = \{e\}$*, an arbitrary singleton,*
- $\mathcal{M}_{\lambda x:A\ t} = \mathcal{M}_t$*,*
- $\mathcal{M}_{(t\ u)} = \mathcal{M}_t$*,*
- $\mathcal{M}_{\Pi x:A\ B}$ *is the set of functions* f *from* $\mathcal{M}_A$ *to* $\mathcal{M}_B$*, except if* $\mathcal{M}_B = \{e\}$*, in which case* $\mathcal{M}_{\Pi x:A\ B} = \{e\}$*, or if* $\mathcal{M}_A = \{e\}$ *in which case* $\mathcal{M}_{\Pi x:A\ B} = \mathcal{M}_B$*.*

Note that, like in the $\lambda\Pi$-calculus and in the $\lambda\Pi$-calculus modulo theory, if t is an object, we have $\mathcal{M}_t = \{e\}$.

Like in the $\lambda\Pi$-calculus, and unlike in the $\lambda\Pi$-calculus modulo theory, if t is a type, we have $\mathcal{M}_t = \{e\}$.

Unlike in the $\lambda\Pi$-calculus and like in the $\lambda\Pi$-calculus modulo theory, if t is a kind we do not have $\mathcal{M}_t = \mathcal{C}$. For instance, $\mathcal{M}_{Type \to Type} = \mathcal{C} \to \mathcal{C}$.

Proposition 12.1 *If* t *and* u *are two well-typed tems in a context* Γ *and* $t \equiv u$ *then* $\mathcal{M}_t = \mathcal{M}_u$*.*

Proof. First note that if u is an object or a type, then $\mathcal{M}_{(u/x)t} = \mathcal{M}_t$. Thus $\mathcal{M}_{((\lambda x:A\ t)\ u)} = \mathcal{M}_t = \mathcal{M}_{(u/x)t}$. We conclude using confluence and subject reduction.

Definition 12.3 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a well-formed context. A Γ-valuation ϕ is a function mapping every variable x_i to an element of $\mathcal{M}_{A_i}$.*

Definition 12.4 *Let t be a term of the Calculus of constructions that has type A in a context Γ or that is equal to $Kind$. Let ϕ be a Γ-valuation. The element $[\![t]\!]_\phi$ of $\mathcal{M}_A$ is defined as follows.*

- *$[\![Type]\!]_\phi$ is the set of strongly terminating terms,*
- *$[\![Kind]\!]_\phi$ is the set of strongly terminating terms,*
- *$[\![x]\!]_\phi = \phi(x)$,*
- *$[\![\lambda x : C\ t]\!]_\phi$ is the function of domain $\mathcal{M}_C$ mapping a in $\mathcal{M}_C$ to $[\![t]\!]_{\phi,x=a}$, except if $[\![t]\!]_{\phi,x=a} = e$ for all a, in which case $[\![\lambda x : C\ t]\!]_\phi = e$, or if $\mathcal{M}_C = \{e\}$, in which case $[\![\lambda x : C\ t]\!]_\phi = [\![t]\!]_{\phi,x=e}$,*
- *$[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi([\![u]\!]_\phi)$, except if $[\![t]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = e$, or if $[\![u]\!]_\phi = e$, in which case $[\![(t\ u)]\!]_\phi = [\![t]\!]_\phi$,*
- *$[\![\Pi x : C\ D]\!]_\phi$ is the candidate $\tilde{\Pi}([\![C]\!]_\phi, \{[\![D]\!]_{\phi,x=c} \mid c \in \mathcal{M}_C\})$.*

Proposition 12.2 *If t and u are two well-typed tems in a context Γ and $t \equiv u$ then $[\![t]\!]_\phi = [\![u]\!]_\phi$.*

Proof. Consider a well-typed term of the form $((\lambda x : B\ t)\ u)$. As this term is well-typed, u has type B and thus $[\![u]\!]_\phi \in \mathcal{M}_B$. Let C be the type of t, if $\mathcal{M}_C \neq \{e\}$ then $[\![((\lambda x : B\ t)\ u)]\!]_\phi = [\![t]\!]_{\phi,x=[\![u]\!]_\phi} = [\![(u/x)t]\!]_\phi$. Otherwise, $[\![((\lambda x : B\ t)\ u)]\!]_\phi = e = [\![(u/x)t]\!]_\phi$. We conclude using confluence and subject reduction.

Proposition 12.3 *Let $\Gamma = x_1 : A_1, ..., x_n : A_n$ be a context, ϕ be a Γ-valuation, σ be a substitution mapping every x_i to an element of $[\![A_i]\!]_\phi$ and t a term of type B in Γ. Then $\sigma t \in [\![B]\!]_\phi$. Hence t strongly terminates for β-reduction.*

Proof. Same as that of in Proposition 10.18.

12.2 Pure Type Systems

The difference between the Simply typed λ-calculus, the $\lambda\Pi$-calculus, and the Calculus of constructions is in the *Product* rules and the *Abstraction* rule. All the Product rules have the same form

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A\ B : s_3}$$

Only the symbols s_1, s_2, s_3 are different. With $\langle s_1, s_2, s_3\rangle = \langle Type, Type, Type\rangle$ we have the base rule that permits to form functional types. The rule $\langle Type, Kind, Kind\rangle$

permits to form dependent types, the rule $\langle Kind, Type, Type\rangle$ polymorphic types, and the rule $\langle Kind, Kind, Kind\rangle$ type constructors.

More generally, we can introduce an arbitrary set S of constants called *sorts*—for example, the set $\{Type, Kind\}$—, an arbitrary set Ax of pairs of sorts called *axioms*—for example, the set $\{\langle Type, Kind\rangle\}$—,and an arbitrary set of triples R of sorts called *rules* and consider the λ-calculus defined by the following rules.

Definition 12.5 (Pure Type Systems)
The empty context is well-formed:

$$\frac{}{[\,]\ \textit{well-formed}}$$

Declaration of a variable:

$$\frac{\Gamma \vdash A : s}{\Gamma, x : A\ \textit{well-formed}}\, s \in S$$

Typing the sorts:

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma \vdash s_1 : s_2}\, \langle s_1, s_2\rangle \in Ax$$

Variables are terms:

$$\frac{\Gamma\ \textit{well-formed} \quad x : A \in \Gamma}{\Gamma \vdash x : A}$$

Product:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A\ B : s_3}\, \langle s_1, s_2, s_3\rangle \in R$$

Abstraction:

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}\, \langle s_1, s_2, s_3\rangle \in R$$

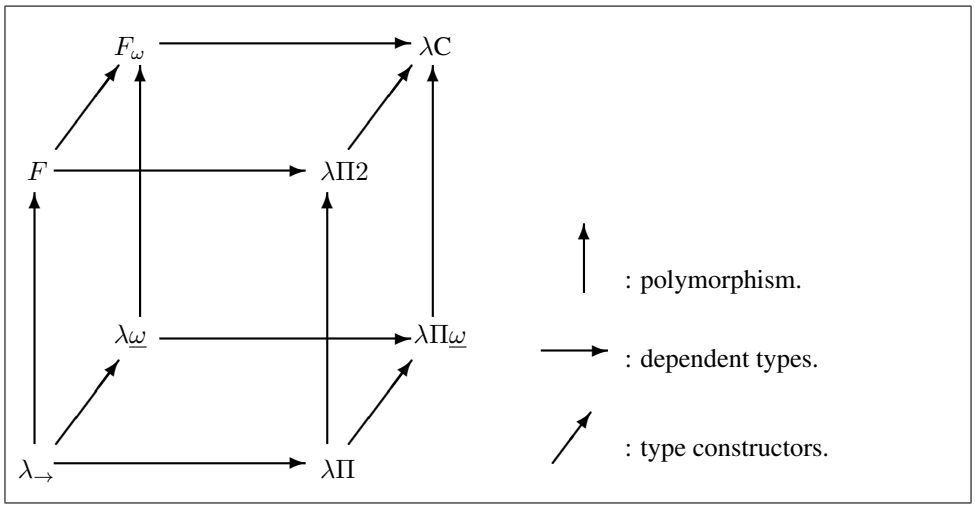
Application:

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : (t'/x)B}$$

Conversion:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : s \quad \Gamma \vdash B : s \quad A \equiv B}{\Gamma \vdash t : B}\, s \in S$$

By taking $S = \{Type, Kind\}$, $Ax = \{\langle Type, Kind\rangle\}$ and R be a set containing the rule $\langle Type, Type, Type\rangle$ and some of the rules $\langle Type, Kind, Kind\rangle$, $\langle Kind, Type, Type\rangle$, $\langle Kind, Kind, Kind\rangle$, we get eight systems, called the systems of Barendregt's cube.



All these systems are subsystem of the Calculus of constructions, thus all terms terminate in all of these systems.

12.3 The System F

The System F is a system of the cube with polymorphic types, and without dependent types and type constructors.

Definition 12.6 (System F (as a Pure Type System))
The empty context is well-formed:

$$\frac{}{[\,] \textit{ well-formed}}$$

Declaration of a variable:

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A \textit{ well-formed}}$$

$$\frac{\Gamma \vdash A : Kind}{\Gamma, x : A \textit{ well-formed}}$$

Type is a kind:

$$\frac{\Gamma \textit{ well-formed}}{\Gamma \vdash Type : Kind}$$

Variables are terms:

$$\frac{\Gamma \textit{ well-formed} \quad x : A \in \Gamma}{\Gamma \vdash x : A}$$

Product:

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash \Pi x : A\ B : Type}$$

Abstraction:

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}$$

$$\frac{\Gamma \vdash A : Kind \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : \Pi x : A\ B}$$

Application:

$$\frac{\Gamma \vdash t : \Pi x : A\ B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : (t'/x)B}$$

Conversion:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : Type \quad \Gamma \vdash B : Type \quad A \equiv B}{\Gamma \vdash t : B}$$

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash A : Kind \quad \Gamma \vdash B : Kind \quad A \equiv B}{\Gamma \vdash t : B}$$

But the System F has a much simpler definition. Indeed, as a product always has type $Type$, $Type$ is the only term of type $Kind$. Hence, we can drop the symbol $Kind$ and replace all terms of type $Kind$ by $Type$.

As in a product $\Pi x : A\ B$, B always has type $Type$, an application is always an object. Thus no application is a type and a type is either a variable of type $Type$ or a product. Thus, by induction on the structure of types, a type is always irreducible and the conversion rule can be dropped as well.

Finally, if A and B have type $Type$, no variable of type A can occur in B and the type $\Pi x : A\ B$ can be written $A \to B$.

Definition 12.7 (System F)
The empty context is well-formed:

$$\frac{}{[\,]\ \textit{well-formed}}$$

Declaration of a variable:

$$\frac{\Gamma \vdash A : Type}{\Gamma, x : A\ \textit{well-formed}}$$

$$\frac{\Gamma\ \textit{well-formed}}{\Gamma, x : Type\ \textit{well-formed}}$$

Variables are terms:

$$\frac{\Gamma\ \textit{well-formed} \quad x : A \in \Gamma}{\Gamma \vdash x : A}$$

Product:

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type}{\Gamma \vdash A \to B : Type}$$

$$\frac{\Gamma, x : Type \vdash B : Type}{\Gamma \vdash \Pi x : Type\ B : Type}$$

Abstraction:

$$\frac{\Gamma \vdash A : Type \quad \Gamma, x : A \vdash B : Type \quad \Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A\ t : A \to B}$$

$$\frac{\Gamma, x : Type \vdash B : Type \quad \Gamma, x : Type \vdash t : B}{\Gamma \vdash \lambda x : Type\ t : \Pi x : Type\ B}$$

Application:

$$\frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash t' : A}{\Gamma \vdash (t\ t') : B}$$

$$\frac{\Gamma \vdash t : \Pi x : Type\ B \quad \Gamma \vdash t' : Type}{\Gamma \vdash (t\ t') : (t'/x)B}$$

If we compare this definition with the definition of Simply typed λ-calculus (Definition 10.1), we see that the System F extends Simply typed λ-calculus by adding three rules. An extra type formation rule, allowing to build types of the form $\Pi x : Type\ A$. For instance $\Pi x : Type\ (x \to x)$, an extra abstraction rule allowing to build terms of such types, for instance the polymorphic identity $\lambda x : Type\ \lambda y : x\ y$, and an extra application allowing to apply such polymorphic functions to a particular type $((\lambda x : Type\ \lambda y : x\ y)\ nat)$.