Notes for the Proof Theory Course

Master 2 "Logique mathématique et fondements de l'informatique", Univ. Paris 7

Damiano Mazza

Contents

1 Natural deduction

We fix an arbitrary first-order language, which induces a set of *terms* (ranged over by $t, u, \ldots$) including first-order variables (ranged over by $x, y, \ldots$) and we denote by $P(x_1, \ldots, x_n)$ a generic atomic formula of such a language containing free occurrences of the variables $x_1, \ldots, x_n$. We shall then consider the formulas generated by the following grammar:

$$A, B ::= P(x_1, \ldots, x_n) \mid A \Rightarrow B \mid A \wedge B \mid A \vee B \mid \forall x.A \mid \exists x.A \mid \bot.$$

We define negation as $\neg A := A \Rightarrow \bot$. *Minimal formulas* are as above but without $\bot$ (and therefore without negation).

The set of free variables of a formula A, denoted by $\mathrm{fv}(A)$, is defined as usual, and we denote by $A[t/x]$ the formula obtained from the capture-free substitution of the term t to all free occurrences of x in the formula A.

We make some notational conventions to avoid too many parentheses in formulas: implication associates to the right, *i.e.*, $A \Rightarrow B \Rightarrow C$ means $A \Rightarrow (B \Rightarrow C)$; and the scope of quantifiers is the largest possible, *i.e.*, $\forall x.A$ means $(\forall x.A)$.

We suppose the reader to be acquainted with classical sequent calculus, or **LK**. From this, the intuitionistic version **LJ** and the minimal version **LM** of sequent calculus may be defined by restricting to sequents of the form $\Gamma \vdash \Delta$ where:

LJ: Δ contains *at most* one formula;

LM: Δ contains *exactly* one formula, the rules for $\bot$ are discarded, and axioms are restricted to minimal formulas.

In natural deduction, proofs consist of trees of the form

$$\begin{array}{ccc} C_1 & \ldots & C_n \\ & \vdots & \\ & A & \end{array}$$

in which the root A is the *conclusion*, *i.e.*, the formula which is proved, and the leaves $C_1, \ldots, C_n$ are the *hypotheses* needed to prove A (they are *occurrences* of formulas, *i.e.*, we may have $C_i = C_j$ for some $i \neq j$). Some leaves of the tree are *discharged* (denoted by $[C_i]$), meaning that they are not to be considered hypotheses. If all leaves are discharged, then the conclusion holds without hypotheses (it will be the case, for example, of every propositional tautology).

1.1 Minimal natural deduction

The proofs of natural deduction for minimal logic (**NM**) are built inductively as follows:

Assumption: for every minimal formula A,

$$A$$

is a proof of conclusion A and hypothesis A.

Implication introduction: if B is provable from an arbitrary number of hypotheses A, then $A \Rightarrow B$ is provable as follows:

$$\dfrac{\begin{array}{c}[A]^* \\ \vdots \\ B\end{array}}{A \Rightarrow B}\ {}^*$$

where the hypotheses A have been "discharged", *i.e.*, they are no longer hypotheses of the proof (observe that some occurrences of A might still be left as hypotheses, and that the rule might actually discharge no hypothesis at all). The sign "$*$" is used to identify which occurrences of A are discharged by the rule.

Implication elimination: if $A \Rightarrow B$ and A are provable, then B is provable (this is *modus ponens*):

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

Conjunction introduction: if A and B are provable, then $A \wedge B$ is provable as follows:

$$\dfrac{A \quad B}{A \wedge B}$$

Conjunction elimination: if $A \wedge B$ is provable, then both A and B are provable, as follows:

$$\dfrac{A \wedge B}{A} \qquad \dfrac{A \wedge B}{B}$$

Disjunction introduction: if any of A or B is provable, then $A \vee B$ is provable, as follows:

$$\dfrac{A}{A \vee B} \qquad \dfrac{B}{A \vee B}$$

Disjunction elimination: if $A \vee B$ is provable, and if C is provable by assuming A and by assuming B, then C is provable:

$$\dfrac{A \vee B \quad \begin{array}{c}[A]^* \\ \vdots \\ C\end{array} \quad \begin{array}{c}[B]^* \\ \vdots \\ C\end{array}}{C}\ {}^*$$

The discharged hypotheses are subjected to the same remarks as in the arrow introduction rule.

Universal quantifier introduction: if A is provable from hypotheses not containing the free variable x, then $\forall x.A$ is provable, under the same hypotheses:

$$\dfrac{A}{\forall x.A}$$

Universal quantifier elimination: if $\forall x.A$ is provable, then all instances of A are provable:

$$\frac{\forall x.A}{A[t/x]}\ t$$

where t is an arbitrary term (which is specified by the rule, in case x does not actually appear in A).

Existential quantifier introduction: if $A[t/x]$ is provable for some term t, then $\exists x.A$ is provable, as follows:

$$\frac{A[t/x]}{\exists x.A}$$

Existential quantifier elimination: if $\exists x.A$ is provable, if $x \notin \mathrm{fv}(C)$ and if from A we may deduce C without additional hypotheses on x, then we may prove C:

$$\frac{\exists x.A \qquad \begin{array}{c}[A]^{*}\\ \vdots\\ C\end{array}}{C}\ {}^{*}$$

The discharged hypotheses are subjected to the same remarks as in the arrow introduction rule.

As examples, here are the proofs of two well known minimal tautologies:[1]

$$\frac{\dfrac{[A]^{*}}{B \Rightarrow A}}{A \Rightarrow B \Rightarrow A}\ {}^{*} \qquad \frac{\dfrac{\dfrac{\dfrac{[A \Rightarrow B \Rightarrow C]^{*} \quad [A]^{\ddagger}}{B \Rightarrow C} \quad \dfrac{[A \Rightarrow B]^{\dagger} \quad [A]^{\ddagger}}{B}}{C}}{A \Rightarrow C}\ {}^{\ddagger}}{(A \Rightarrow B) \Rightarrow A \Rightarrow C}\ {}^{\dagger}}{(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C}\ {}^{*}$$

1.2 Intuitionistic natural deduction

The proofs of natural deduction for intuitionistic logic (**NJ**) are those of **NM** extended as follows:

Assumption: any formula is allowed (not just minimal ones);

Absurdity elimination: if a set of hypotheses led us to a proof of $\bot$, then we may prove anything at all from the same hypotheses (this is *ex falso quodlibet*):

$$\frac{\bot}{A}$$

Note that there is no introduction rule for absurdity.

[1] Every propositional minimal tautology is a consequence of these two formulas via the *modus ponens* rule. Anticipating on Sect. 2, the discovery of what we now call the Curry-Howard correspondence began with Haskell Curry's remark that these two formulas exactly match the types of the two combinators **K** and **S**, which in turn generate every λ-term through application.

1.3 Classical natural deduction

The proofs of natural deduction for classical logic (**NK**) are those of **NJ** with one additional rule:

Double negation elimination: if we have a proof of $\neg\neg A$, then we have a proof of A:

$$\frac{\neg\neg A}{A}$$

Other rules may be considered instead of the above one, such as the following, also known as *reductio ad absurdum* (proof by contradiction):

$$\dfrac{\begin{array}{c}[\neg A]^{*}\\ \vdots\\ \bot\end{array}}{A}\ {}^{*}$$

The usual remarks for discharged formulas apply. Note the difference between the above rule and the application of an implication introduction rule, perfectly valid in intuitionistic logic:

$$\dfrac{\begin{array}{c}[\neg A]^{*}\\ \vdots\\ \bot\end{array}}{\neg\neg A}\ {}^{*}$$

Classical logic comes from intuitionistic logic by identifying the impossibility of invalidity with validity, which is what the double negation elimination rule is about. In fact, we may obtain *reductio* by applying a double negation elimination rule to the above proof. Conversely, from the latter we may straightforwardly obtain double negation elimination:

$$\dfrac{\dfrac{\neg\neg A \quad [\neg A]^{*}}{\bot}}{A}\ {}^{*}$$

(remember that $\neg\neg A$ is defined as $(A \Rightarrow \bot) \Rightarrow \bot$).

A further alternative is to use excluded middle (*tertium non datur*) as an axiom, *i.e.*, a rule with no hypothesis:

$$\frac{}{\neg A \vee A}$$

We may easily prove the equivalence of this rule with the other two introduced above. For instance, this is how we can prove the excluded middle by contradiction:

$$\dfrac{\dfrac{\dfrac{[\neg(\neg A \vee A)]^{*} \quad \dfrac{[\neg A]^{\dagger}}{\neg A \vee A}}{\bot}}{\neg\neg A}\ {}^{\dagger} \qquad \dfrac{\dfrac{[\neg(\neg A \vee A)]^{*} \quad \dfrac{[A]^{\ddagger}}{\neg A \vee A}}{\bot}}{\neg A}\ {}^{\ddagger}}{\bot}}{\neg A \vee A}\ {}^{*}$$

where the last rule is a *reductio ad absurdum*. Conversely, the double negation elimination rule may be derived using excluded middle as follows:

$$\dfrac{\dfrac{}{\neg A \vee A} \qquad \dfrac{\dfrac{\neg\neg A \quad [\neg A]^*}{\bot}}{A} \qquad [A]^*}{A}\ {}^*$$

where the last rule is a disjunction elimination, and we used absurdity elimination (*ex falso quodlibet*) in the left branch.

1.4 Natural deduction and sequent calculus

Provability in natural deduction exactly matches provability in sequent calculus. To prove this formally, it is better to slightly modify the formulation of natural deduction using *judgments*, which look like sequents with exactly one formula on the right:

$$\Gamma \vdash A.$$

For brevity, in this section we shall restrict ourselves to implication and disjunction (and absurdity, for intuitionistic and classical logic). The reader is invited to verify that all results carry over to conjunction and quantifiers.

The implication-disjunction fragment of **NM** may be reformulated as follows:

$$\dfrac{}{\Gamma, A \vdash A}$$

$$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \qquad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \qquad \dfrac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$$

The rules obviously correspond to assumption, introduction and elimination of implication, and introduction and elimination of disjunction.

For **NJ**, apart from allowing non-minimal formulas in the assumption rule, we add the rule

$$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$$

Finally, to obtain **NK**, we further add the rule

$$\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$$

It is fairly straightforward to see that this formulation is equivalent to the one given in terms of trees. In fact, in the tree version, it does not hurt to suppose that assumptions actually introduce a number of "ghost hypotheses", so that implication introduction always discharges at least one hypothesis; similarly, using "ghost hypotheses", we may assume that the rules with more than one premise (implication elimination and disjunction elimination) involve trees sharing exactly the same hypotheses. In this way, we obtain the alternative formulation given above.

Proposition 1 *If* $\Gamma \vdash A$ *is provable in* **NM** *(resp.* **NJ***,* **NK***) then* $\Gamma \vdash A$ *is provable in* **LM** *(resp.* **LJ***,* **LK***).*

Proof. By induction on the last rule of the proof of $\Gamma \vdash A$ in natural deduction. The assumption rule and the introduction rules are immediate. For what concerns the elimination rules, we have (implicitly using the induction hypothesis)

$$\dfrac{\dfrac{\vdots\,1}{\Gamma \vdash B \Rightarrow C} \quad \dfrac{\vdots\,2}{\Gamma \vdash B}}{\Gamma \vdash C} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots\,1}{\Gamma \vdash B \Rightarrow C} \quad \dfrac{\dfrac{\vdots\,2}{\Gamma \vdash B} \quad \dfrac{}{C \vdash C}}{\Gamma, B \Rightarrow C \vdash C}}{\dfrac{\Gamma, \Gamma \vdash C}{\Gamma \vdash C}}$$

and

$$\dfrac{\dfrac{\vdots\,1}{\Gamma \vdash B \vee C} \quad \dfrac{\vdots\,2}{\Gamma, B \vdash A} \quad \dfrac{\vdots\,3}{\Gamma, C \vdash A}}{\Gamma \vdash A} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots\,1}{\Gamma \vdash B \vee C} \quad \dfrac{\dfrac{\vdots\,2}{\Gamma, B \vdash A} \quad \dfrac{\vdots\,3}{\Gamma, C \vdash A}}{\Gamma, B \vee C \vdash A}}{\dfrac{\Gamma, \Gamma \vdash A}{\Gamma \vdash A}}$$

In both cases, an elimination rule is translated into a left rule (the additive version in the case of disjunction), followed by a cut rule, followed by structural rules (*i.e.*, contractions) if necessary.

In the case of **NJ**, we also need to treat the elimination of absurdity. The pattern is the same (left rule, cut rule, structural rules, a weakening in this case):

$$\dfrac{\dfrac{\vdots\,1}{\Gamma \vdash \bot}}{\Gamma \vdash A} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\vdots\,1}{\Gamma \vdash \bot} \quad \dfrac{}{\bot \vdash}}{\Gamma \vdash}}{\Gamma \vdash A}$$

For **NK**, translating double negation elimination is immediate:

$$\dfrac{\dfrac{\vdots\,1}{\Gamma \vdash \neg\neg A}}{\Gamma \vdash A} \quad \rightsquigarrow \quad \dfrac{\dfrac{\vdots\,1}{\Gamma \vdash \neg\neg A} \quad \dfrac{\dfrac{}{A \vdash A}}{\dfrac{\vdash \neg A, A}{\neg\neg A \vdash A}}}{\Gamma \vdash A}$$

Observe that the left and right negation rules used above are derivable in **LK** using left and right rules for $\bot$ and implication, respectively (remember that negation is not a connective, $\neg A$ is defined as $A \Rightarrow \bot$):

$$\dfrac{\Gamma \vdash \Delta, A \quad \dfrac{}{\bot \vdash}}{\Gamma, \neg A \vdash \Delta} \qquad \text{and} \qquad \dfrac{\dfrac{\Gamma, A \vdash \Delta}{\Gamma, A \vdash \bot, \Delta}}{\Gamma \vdash \neg A, \Delta}$$

□

For the converse, we need an auxiliary lemma to deal with structural rules. In the following, by "proof" we mean "natural deduction proof".

Lemma 2 (Cut and structural rules) *1. If π and σ are proofs of $\Gamma \vdash A$ and $\Delta, A \vdash B$, respectively, then there is a proof of $\Gamma, \Delta \vdash B$ using exactly the logical rules of π and σ;*

2. if π is a proof of $\Gamma \vdash C$, then $\Gamma, A \vdash C$ is provable using the same logical rules as π;

3. if π is a proof of $\Gamma, A, A \vdash C$, then $\Gamma, A \vdash C$ is provable using the same logical rules as π.

Proof. All points are proved by straightforward inductions. For point 1, the tree formulation gives a quite convincing intuition: from

$$\begin{array}{c}\Gamma\\ \vdots\ \pi\\ A\end{array} \qquad \text{and} \qquad \begin{array}{c}\Delta\quad A\\ \quad\ \vdots\ \sigma\\ \quad\ B\end{array}$$

we construct

$$\begin{array}{c}\quad\ \Gamma\\ \quad\ \vdots\ \pi\\ \Delta\quad A\\ \quad\ \vdots\ \sigma\\ \quad\ B\end{array}$$

Intuitively, the other two points hold because we may add (or remove) hypotheses in assumption rules as we please. □

Let Δ be a sequence of formulas $D_1, \ldots, D_n$. In the following, we set $\bigvee \Delta = D_1 \vee \ldots \vee D_n$ (we fix one associative pattern, they are all provably equivalent anyway), and $\bigvee \Delta = \bot$ in case $n = 0$.

Proposition 3 *If $\Gamma \vdash \Delta$ is provable in* **LM** *(resp.* **LJ**, **LK***), then $\Gamma \vdash \bigvee \Delta$ is provable in* **NM** *(resp.* **NJ**, **NK***).*

Proof. The proof is once more by induction on the last rule proving the sequent $\Gamma \vdash \Delta$. The case of **LM** is simpler because the sequent is actually of the form $\Gamma \vdash D$, with D a minimal formula. As usual, the axiom rule and the right rules are immediate. The cut rule and the structural rules on the left are given by Lemma 2. For what concerns the left rules, we have, using the induction hypothesis and point 1 of Lemma 2,

$$\dfrac{\begin{array}{cc}\vdots\ 1 & \vdots\ 2\\ \Gamma \vdash A & \Sigma, B \vdash D\end{array}}{\Gamma, \Sigma, A \Rightarrow B \vdash D} \quad \rightsquigarrow \quad \dfrac{\dfrac{}{\Gamma, A \Rightarrow B \vdash A \Rightarrow B} \quad \begin{array}{c}\vdots\ 1\\ \Gamma \vdash A\end{array}}{\begin{array}{c}\Gamma, A \Rightarrow B \vdash B\\ \vdots\ 2\\ \Gamma, \Sigma, A \Rightarrow B \vdash D\end{array}}$$

and, using the induction hypothesis and point 2 of Lemma 2,

$$\dfrac{\begin{array}{cc}\vdots\ 1 & \vdots\ 2\\ \Gamma, A \vdash D & \Gamma, B \vdash D\end{array}}{\Gamma, A \vee B \vdash D} \quad \rightsquigarrow$$

$$\dfrac{\dfrac{}{\Gamma, A \vee B \vdash A \vee B} \quad \Gamma, A \vee B, A \vdash D \overset{1}{\vdots} \quad \Gamma, A \vee B, B \vdash D \overset{2}{\vdots}}{\Gamma, A \vee B \vdash D}$$

So left rules are translated, as expected, as elimination rules (plus some structural manipulations).

The translation of **LJ** is immediate: the left $\bot$ rule is just an assumption rule in **NJ**, and the right rule may only be applied when Δ is empty, so it translates to nothing in **NJ**, because in that case we already have $\bigvee \Delta = \bot$.

The case of **LK** requires some additional work, because right rules may result in sequents with more than one formula to the right, *i.e.*, in Δ, and this has non-trivial consequences with our definition of $\bigvee \Delta$. First we use the fact that, in a proof of $\Gamma \vdash \bigvee \Delta$, we may always "single out" an element of the disjunction:

Lemma 4 *In* **NJ**, $\Gamma \vdash D \vee A$ *provable implies* $\Gamma, \neg D \vdash A$ *provable.*

Proof. By point 2 of Lemma 2, if we have $\Gamma \vdash D \vee A$ we also have $\Gamma, \neg D \vdash D \vee A$. Then, using a disjunction elimination rule, we have

$$\dfrac{\Gamma, \neg D \vdash D \vee A \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, \neg D, D \vdash \neg D} \quad \dfrac{}{\Gamma, \neg D, D \vdash D}}{\Gamma, \neg D, D \vdash \bot}}{\Gamma, \neg D, D \vdash A} \qquad \dfrac{}{\Gamma, \neg D, A \vdash A}}{\Gamma, \neg D \vdash A}$$

□

Then, we use the fact that, in classical logic, $D \Rightarrow A$ is equivalent to $\neg D \vee A$, which is subsumed by the following:

Lemma 5 *In* **NK**, *we have* $\neg D \Rightarrow A \vdash D \vee A$.

Proof. We give the derivation in tree form, which is much more compact:

$$\dfrac{\dfrac{\dfrac{[\neg(D \vee A)]^{\ddagger} \quad \dfrac{[A]^{\dagger}}{D \vee A}}{\dfrac{\bot}{\neg A}\,{\scriptstyle \dagger}} \qquad \dfrac{\neg D \Rightarrow A \qquad \dfrac{\dfrac{[\neg(D \vee A)]^{\ddagger} \quad \dfrac{[D]^{*}}{D \vee A}}{\bot}}{\neg D}\,{\scriptstyle *}}{A}}{\bot}}{\dfrac{\neg\neg(D \vee A)}{D \vee A}\,{\scriptstyle \ddagger}}$$

Note the (fundamental!) use of double negation elimination. □

The proof now is easy: right weakening, disjunction and $\bot$ rules are immediate (modulo associativity of the disjunction $\bigvee \Delta$). For what concerns right contraction and right implication, the strategy is the same: we single out the active occurrence(s) through Lemma 4; then, we apply the appropriate rule and we use Lemma 5 to get the desired conclusion. We give the detail of the right implication rule, leaving contraction to the reader. We want to simulate the rule

$$\dfrac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \Rightarrow B.}$$

By the induction hypothesis, in **NK** we have $\Gamma, A \vdash \bigvee \Delta \vee B$ (perhaps modulo some re-association of the disjunctions). Using Lemma 4, we obtain a proof of $\Gamma, \neg \bigvee \Delta, A \vdash B$, from which we build

$$\dfrac{\dfrac{\Gamma, \neg \bigvee \Delta, A \vdash B}{\Gamma, \neg \bigvee \Delta \vdash A \Rightarrow B}}{\Gamma \vdash \neg\Delta \Rightarrow (A \Rightarrow B).}$$

From that, Lemma 5 gives us the desired proof of $\Gamma \vdash \bigvee \Delta \vee (A \Rightarrow B)$ (via point 1 of Lemma 2). □

1.5 Normalization (cut-elimination in natural deduction)

Let us look at the translation in **NM** of a cut rule whose both premises are introduced by the rules immediately above (a "key case" of cut-elimination). The most important case for our purposes is that in which the cut formula is of the form $A \Rightarrow B$. We furthermore suppose that the occurrence of A used by the right rule is the result of a certain number of structural rules (contractions or weakening):

$$\dfrac{\dfrac{\dfrac{\vdots\ 1}{\dfrac{\Gamma, A, \ldots, A \vdash B}{\Gamma, A \vdash B}}}{\Gamma \vdash A \Rightarrow B} \qquad \dfrac{\vdots\ 2}{\Delta \vdash A} \quad \dfrac{\vdots\ 3}{B, \Sigma \vdash C}}{\Gamma, \Delta, \Sigma \vdash C}$$

The **NM** translation is

$$\dfrac{\Gamma \quad \dfrac{\dfrac{[A]^* \ldots [A]^*}{\vdots\ 1}}{B}}{\dfrac{A \Rightarrow B \quad {}^*}{B}} \qquad \dfrac{\Delta}{\vdots\ 2} \quad \Sigma}{\dfrac{A}{\vdots\ 3}}{C}$$

We notice that *the introduction of the implication $A \Rightarrow B$ is immediately followed by its elimination.* The reader is invited to check that such an introduction/elimination pattern is present in the **NM** translation of all "key cases" of the cut rule in **LM**.

The introduction/elimination pattern looks unnecessary: in the above case, we could eliminate it by considering the proof

$$\dfrac{\Gamma \quad \dfrac{\dfrac{\dfrac{\Delta}{\vdots\ 2}}{A} \quad \ldots \quad \dfrac{\dfrac{\Delta}{\vdots\ 2}}{A}}{\vdots\ 1}}{B} \qquad \Sigma}{\dfrac{\vdots\ 3}{C}}$$

Note that the proof marked by (2) may be duplicated several times, or completely discarded, according to the number of occurrences of A discharged by the introduction rule under consideration.

The reader is invited to check that, if one translates in **NM** the reduced version of the "key case", which is

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots\ 2}{\Delta \vdash A} \quad \dfrac{\vdots\ 1}{\Gamma, A, \ldots, A \vdash B}}{\dfrac{\vdots\ 2}{\Delta \vdash A} \quad \dfrac{\vdots}{\Gamma, \Delta, \ldots, A \vdash B}}}{\dfrac{\Gamma, \Delta, \ldots, \Delta \vdash B}{\Gamma, \Delta \vdash B}} \quad \dfrac{\vdots\ 3}{B, \Sigma \vdash C}}{\Gamma, \Delta, \Sigma \vdash C}$$

(where all rules are either cuts or structural), one obtains exactly the "simplified" natural deduction proof given above. In other words, the obvious way of getting rid of the introduction/elimination pattern precisely matches the standard cut-elimination step.

For this reason, the natural deduction equivalent of a cut-free proof is a proof containing no introduction/elimination pattern as the one above. Such proofs are called *normal*. Accordingly, the cut-elimination process is replaced by a *normalization* process, which aims at eliminating all introduction/elimination patterns by means of the following rewriting rules:

$$\dfrac{\dfrac{\begin{array}{c}[A]^* \ldots [A]^* \\ \vdots \\ B\end{array}}{A \Rightarrow B}\ {}^{*} \quad \dfrac{\vdots}{A}}{B} \quad \rightarrow \quad \dfrac{\dfrac{\vdots}{A} \ldots \dfrac{\vdots}{A}}{\vdots\ B}$$

$$\dfrac{\dfrac{\dfrac{\vdots}{A} \quad \dfrac{\vdots}{B}}{A \wedge B}}{A} \quad \rightarrow \quad \dfrac{\vdots}{A} \qquad\qquad \dfrac{\dfrac{\dfrac{\vdots}{A} \quad \dfrac{\vdots}{B}}{A \wedge B}}{B} \quad \rightarrow \quad \dfrac{\vdots}{B}$$

$$\dfrac{\dfrac{\vdots\ \pi}{A}}{\dfrac{\forall x.A}{A[t/x]}\ {}^{t}} \quad \rightarrow \quad \dfrac{\vdots\ \pi[t/x]}{A[t/x]}$$

In the last rule, by $\pi[t/x]$ we mean the whole proof π in which every free occurrence of x is replaced by t in every formula occurring in π. This is possible because, thanks to the condition on the introduction rule of universal quantification, x does not appear free in the hypotheses of π.

Of course, it is implicitly intended that the above rewriting rules may be applied at any place within a proof, not just at the root.

If we restrict to the fragment of **NM** containing only implication, conjunction and universal quantification (the *negative fragment*), the above rules are enough. This is in contrast with cut-elimination in sequent calculus, which

needs a great number of so-called *commuting conversions* in addition to the steps dealing with the "key cases". On the other hand, if we wish to consider also the positive connectives (disjunction and existential quantification), then the reduction steps corresponding to the "key cases" must also be accompanied by commuting conversions. For instance, in the case of disjunction, we have the reduction

$$\dfrac{\dfrac{\vdots}{A}}{A \vee B} \quad \begin{matrix}[A]^* \ldots [A]^* \\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B]^* \ldots [B]^* \\ \vdots \\ C\end{matrix} \quad {}^*_{C} \qquad \rightarrow \qquad \begin{matrix}\vdots & \vdots \\ A & A \\ & \vdots \\ & C\end{matrix}$$

and its symmetric counterpart

$$\dfrac{\dfrac{\vdots}{B}}{A \vee B} \quad \begin{matrix}[A]^* \ldots [A]^* \\ \vdots \\ C\end{matrix} \quad \begin{matrix}[B]^* \ldots [B]^* \\ \vdots \\ C\end{matrix} \quad {}^*_{C} \qquad \rightarrow \qquad \begin{matrix}\vdots & \vdots \\ B & B \\ & \vdots \\ & C\end{matrix}$$

but we also have 6 commuting conversions, one for each elimination rule, such as

$$\dfrac{A \vee B \quad \begin{matrix}[A]^* \\ \vdots \\ C \wedge D\end{matrix} \quad \begin{matrix}[B]^* \\ \vdots \\ C \wedge D\end{matrix}}{\dfrac{C \wedge D}{C}}{}^* \qquad \rightarrow \qquad \dfrac{A \vee B \quad \dfrac{\begin{matrix}[A]^* \\ \vdots \\ C \wedge D\end{matrix}}{C} \quad \dfrac{\begin{matrix}[B]^* \\ \vdots \\ C \wedge D\end{matrix}}{C}}{C}{}^*$$

If we also consider existential quantification, then there are a total of 14 commuting conversions (7 for disjunction, 7 for existential quantification). These are all needed to achieve a normal form because, without them, the introduction/elimination patterns which form "across" positive elimination rules, and which still correspond to cuts in sequent calculus, would be impossible to remove.

2 The Curry-Howard correspondence

2.1 The simply typed λ-calculus

We fix a (denumerably infinite) set of *ground types* $X, Y, \ldots$ (which can be thought of as including, for instance, the type of Booleans, integers, lists of integers, etc.). *Simple types* are generated by the following grammar:

$$T, U ::= X \mid T \to U,$$

where X ranges over ground types.

We fix a denumerably infinite set of *variables* $x, y, \ldots$ (in spite of the identical notation, these are not to be confused with the first-order variables used in a first-order language). Additionally, we fix a function $\mathcal{T}$ from variables to types such that, for every type T, there are infinitely many variables x such that $\mathcal{T}(x) = T$.

Logic		Computer Science
formula	=	type
proof	=	program
cut-elimination	=	execution

Table 1: The Curry-Howard correspondence

The *simply-typed terms*, together with their assigned type, are inductively defined as follows. In the definition, we use the notation $M : T$ as an abbreviation for "M is a term of type T" and we also use the notation $\mathrm{fv}(M)$ to denote the set of free variables of a term M, defined at the same time as the terms themselves, in the usual way.

Variable: for every variable x such that $\mathcal{T}(x) = T$, $x^T : T$, and $\mathrm{fv}(x^T) = \{x\}$.

Abstraction: for every variable x such that $\mathcal{T}(x) = T$ and $M : U$, $\lambda x^T.M : T \to U$, and $\mathrm{fv}(\lambda x^T.M) = \mathrm{fv}(M) \setminus \{x\}$.

Application: for all $M : T \to U$ and $N : T$, $MN : U$, and $\mathrm{fv}(MN) = \mathrm{fv}(M) \cup \mathrm{fv}(N)$.

To make notations lighter, we stipulate that application associates to the left, *i.e.*, KMN means $(KM)N$, and we drop the type superscripts for bound variables, since their type is already specified in the abstraction: $\lambda x^T.x$ means $\lambda x^T.x^T$.

Computation in the λ-calculus is expressed by the following rewriting rule, called β-*reduction*, which is easily seen to preserve types:

$$(\lambda x^T.M)N \quad \to \quad M[N/x]$$

where $M[N/x]$ denotes, as usual, the capture-free substitution of N to all free occurrences of x appearing in M. Just like in the pure λ-calculus, a term of the form given in the left-hand side of the above rules is called *redex*. Of course, the above rewriting rules are intended to apply to any subterm which is a redex. A term containing no redex is called *normal*.

We may now observe that the simple types are in bijection with the formulas of the implication fragment of minimal logic, with the "dictionary"

- ground types $\longleftrightarrow$ atomic formulas;
- arrow type $\longleftrightarrow$ implication.

Similarly, the simply-typed terms are in bijection with the proofs of the implication fragment of **NM**, with the "dictionary":

- variables $\longleftrightarrow$ assumptions;
- abstraction $\longleftrightarrow$ implication introduction rule;
- application $\longleftrightarrow$ implication elimination rule.

Moreover, and this is the key observation, β-reduction exactly matches the normalization rule of **NM**, *i.e.*, we have

- one β-reduction step $\longleftrightarrow$ one normalization step,

as we invite the reader to check. This is the *Curry-Howard correspondence*, summarized in Table 1.

2.2 Product and sum types

Now that we know that each formula is a type and each proof is a program, we may wonder whether the remaining logical constructions (conjunction, disjunction, quantifiers) have any computational meaning. It turns our that the answer is positive, with the exception of first-order quantifiers, which do not give anything interesting. Second-order quantifiers, on the contrary, are extremely interesting from a computational point of view, as we shall see later.

A *product type* is of the form $T \times U$, where T and U are types. The corresponding constructions on terms are

Pairing: if $M : T$ and $N : U$, then $\langle M, N\rangle : T \times U$;

Projections: if $M : T \times U$, then $\pi_1 M : T$ and $\pi_2 M : U$.

Pairing and projections interact through the following rewriting rule, which is added to β-reduction:

$$\pi_1\langle M, N\rangle \quad \to \quad M \qquad\qquad \pi_2\langle M, N\rangle \quad \to \quad N$$

A *sum type* is of the form $T+U$, where T and U are types. The corresponding constructions on terms are

Injections: if $M : T$ and $N : U$, then $\iota_1^U M : T + U$ and $\iota_2^T N : T + U$;

Case: if $M : T + U$ and $P, Q : V$, then $\mathsf{case}(x^T, y^U)\, M\, P\, Q : V$, and x (resp. y) becomes bound in P (resp. Q).

In presence of injections and case constructs, β-reduction is augmented with the following rewriting rules:

$$\mathsf{case}(x^T, y^U)\, (\iota_1^U M)\, P\, Q \quad \to \quad P[M/x]$$

$$\mathsf{case}(x^T, y^U)\, (\iota_2^T M)\, P\, Q \quad \to \quad Q[M/y]$$

This time, we also need the following commuting conversions:

$$(\mathsf{case}(x^T, y^U)\, M\, P\, Q)N \quad \to \quad \mathsf{case}(x^T, y^U)\, M\, (PN)\, (QN)$$

$$\pi_i(\mathsf{case}(x^T, y^U)\, M\, P\, Q) \quad \to \quad \mathsf{case}(x^T, y^U)\, M\, (\pi_i P)\, (\pi_i Q)$$

$$\mathsf{case}(x^T, y^U)\, (\mathsf{case}(z^V, w^W)\, M\, R\, S)\, P\, Q \quad \to$$

$$\mathsf{case}(z^V, w^W)\, M\, (\mathsf{case}(x^T, y^U)\, R\, P\, Q)\, (\mathsf{case}(x^T, y^U)\, S\, P\, Q)$$

We invite the reader to check that the Curry-Howard correspondence extends to product and sum types, with the following additions to the "dictionary":

- product type $\longleftrightarrow$ conjunction;
- sum type $\longleftrightarrow$ disjunction;

- pairing $\longleftrightarrow$ conjunction introduction rule;
- projections $\longleftrightarrow$ conjunction elimination rules;
- injections $\longleftrightarrow$ disjunction introduction rules;
- case $\longleftrightarrow$ disjunction elimination rule;

and, of course, the augmented β-reduction still corresponds to normalization.

3 System F

3.1 Intuitionistic second-order propositional logic

The formulas of (the implication fragment of) second-order propositional logic are defined by the following grammar:

$$A, B ::= X \mid A \Rightarrow B \mid \forall X.A,$$

where X ranges over a denumerably infinite set of *propositional atoms*.

The set of free propositional atoms of a formula A is defined as usual, and is denoted by $\mathrm{fv}(A)$. If A, B are formulas, $A[B/X]$ denotes, as usual, the capture-free substitution of B to all free occurrences of X in A.

The proofs of intuitionistic second-order propositional logic are those obtained using the rules of minimal natural deduction, in which second-order quantification is treated as follows:

Universal quantifier introduction: if A is provable from hypotheses not containing the free variable X, then $\forall X.A$ is provable, under the same hypotheses:

$$\frac{A}{\forall X.A}$$

Universal quantifier elimination: if $\forall X.A$ is provable, then all instances of A are provable:

$$\frac{\forall X.A}{A[B/X]}\ B$$

where B is an arbitrary formula (which is specified by the rule, in case X does not actually appear in A).

Normalization for second order quantification is very similar to the first order case:

$$\frac{\dfrac{\begin{array}{c}\vdots\ \pi\\ A\end{array}}{\forall X.A}}{A[B/X]}\ B \qquad \rightsquigarrow \qquad \begin{array}{c}\vdots\ \pi[B/X]\\ A[B/X]\end{array}$$

where $\pi[B/X]$ denotes the proof π in which every occurrence of the atom X has been replaced by the formula B (this is why B must be specified by the rule itself).

Observe that, with second-order quantification, minimal natural deduction is actually as expressive as intuitionistic natural deduction, because absurdity may be defined as $\bot := \forall X.X$. In fact, every other usual logical connective is also definable in terms of implication and second-order universal quantification:

- $A \wedge B := \forall X.(A \Rightarrow B \Rightarrow X) \Rightarrow X$;
- $A \vee B := \forall X.(A \Rightarrow X) \Rightarrow (B \Rightarrow X) \Rightarrow X$;
- $\exists X.A := \forall Y.(\forall X.A \Rightarrow Y) \Rightarrow Y$.

We invite the reader to check that, with the above definitions, the usual introduction and elimination rules for the respective connectives are derivable, and that the derivations satisfy the appropriate normalization rules for eliminating introduction/elimination patterns.

Hence, the system we are dealing with is actually second order propositional **NJ**, which is equivalent, in terms of provability, to second order propositional **LJ**, thanks to Propositions 1 and 3, which immediately carry over to second order.

3.2 Polymorphic types

The *types* of **F** are the simple types plus the *polymorphic types*:

$$T, U ::= X \mid T \to U \mid \Pi X.T.$$

Free and bound type variables are defined as usual, with ΠX being a binder.

The *terms* of **F**, together with their assigned type, are inductively defined by adding the two following formation rules to those of the simply typed λ-calculus:

Universal abstraction: for every propositional atom X and $M : T$, $\Lambda X.M : \Pi X.T$, provided that, for all $x \in \mathrm{fv}(M)$, $X \notin \mathrm{fv}(\mathcal{T}(x))$. In that case, $\mathrm{fv}(\Lambda X.M) = \mathrm{fv}(M)$.

Universal application: for all $M : \forall X.T$ and type U, $MU : T[U/X]$, and $\mathrm{fv}(MU) = \mathrm{fv}(M)$.

Computation in system **F** is expressed by usual β-reduction plus the following rewriting rule (which is, of course, applicable in any context):

$$(\Lambda X.M)T \quad \to \quad M[T/X],$$

where $M[T/X]$ denotes the capture-free substitution of T to all free occurrences of X appearing in M. This rule allows types to change dynamically, which is the meaning of polymorphism.

Observe that the second reduction rule is isomorphic to the normalization rule for second order quantification in **NJ**. Therefore, system **F** extends the Curry-Howard correspondence, by adding the following entries to our "dictionary":

- universal abstraction $\longleftrightarrow$ universal quantifier introduction rule;
- universal application $\longleftrightarrow$ universal quantifier elimination rule.

One consequence of this is that product and sum types may be defined in system **F**, using the second-order definition of conjunction and disjunction. For instance, for product types, we have

$$T \times U := \Pi X.(T \to U \to X) \to X,$$

and

$$\langle M, N\rangle := \Lambda X.\lambda c^{T\to U\to X}.cMN,$$

$$\pi_1^{U,V} := \lambda p^{T\times U}.pT(\lambda x^T y^U.x), \qquad \pi_2^{U,V} := \lambda p^{T\times U}.pU(\lambda x^T y^U.y).$$

These terms mirror exactly the derivations of the second-order encoding of introduction and elimination rules for conjunction. The fact that the encoding respects the usual normalization rules of natural deduction allows us to conclude immediately that

$$\pi_1^{T,U}\langle M, N\rangle \to^* M, \qquad \pi_2^{T,U}\langle M, N\rangle \to^* N,$$

as expected. Of course, we invite the reader to do the computation in system **F** and check this, also in the case of disjunction/sum types.

4 Programming in system F

4.1 Free structures

Much more than product and sum types can be encoded in system **F**. In fact, any *free structure* (or *free algebra*, as it is often called in computer science) has a natural representation in system **F**.

The most general way of defining free structures uses category theory and the concept of *initial algebra*. We recall the definition.

Definition 1 (Initial algebra) *Let $\mathcal{A}$ be a category, and let $F : \mathcal{A} \to \mathcal{A}$ be an endofunctor. An* F-algebra *is a pair (A, α) where A is an object of $\mathcal{A}$ and $\alpha : FA \to A$. Given two F-algebras $(A, \alpha), (B, \beta)$, an F-morphism between them is a morphism $f : A \to B$ which makes the following diagram commute:*

$$\begin{array}{ccc} FA & \xrightarrow{\alpha} & A \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle f} \\ FB & \xrightarrow[\beta]{} & B \end{array}$$

It is easy to see that F-algebras and F-morphisms form a category, denoted by $\mathcal{A}_F$. An F-algebra is initial *if it is an initial object in $\mathcal{A}_F$.*

Of course, initial algebras are unique modulo a unique isomorphism, so we may speak of *the* initial algebra of a functor. The initial algebra of F is the "smallest" fixpoint of F, in the following sense:

Proposition 6 *If (Z, ζ) is the initial algebra of $F : \mathcal{A} \to \mathcal{A}$, then $(FZ, F\zeta) \cong (Z, \zeta)$ in $\mathcal{A}_F$.*

Proof. Note that, for every F-algebra (A, α), $(FA, F\alpha)$ is also an F-algebra, and it is immediate to see that α is an F-morphism from $(FA, F\alpha)$ to (A, α). In the case of (Z, ζ), its initial property gives us a (unique) F-morphism $\iota : (Z, \zeta) \to (FZ, F\zeta)$. Again using the initial property, we have $\zeta \circ \iota = id_Z$. For what concerns the opposite direction, using the fact that ι is an F-morphism and that F is a functor, we have $\iota \circ \zeta = F\zeta \circ F\iota = F(\zeta \circ \iota) = id_{FZ}$. □

In the following, we denote by $A+B$ the disjoint union of two sets A and B, and we denote by $[f,g]$ the copairing of two functions $f : A \to C$, $g : B \to C$, which is the canonical function of type $A + B \to C$ defined "case-wise from f and g. Of course the copairing notation generalizes to an arbitrary number of functions: for any family of sets $(A_i)_{i\in I}$, if $f_i : A_i \to C$, $[f_i]_{i\in I} : \sum_{i\in I} A_i \to C$.

Definition 2 (Free structure) *Let $C_1, \ldots, C_n$: **Set** $\to$ **Set** be functors. A* free structure *on $C_1, \ldots, C_n$ is a tuple $(\Theta, c_1, \ldots, c_n)$ where Θ is a set and $c_i : C_i\Theta \to \Theta$ for $1 \leq i \leq n$, such that $(\Theta, [c_1, \ldots, c_n])$ is the initial algebra of the functor $C_1 + \cdots + C_n$: **Set** $\to$ **Set**.*

The functions $c_1, \ldots, c_n$ are to be seen as constructors, whose types are given by the functors $C_1, \ldots, C_n$. The fact that the algebra is initial for $C_1 + \cdots + C_n$ means, by Proposition 6, that every element of Θ is of the form $c_i(z)$ with $z \in C_i(\Theta)$ for some $1 \leq i \leq n$ and that i and z are unique.

The key consequence is that we may define functions on Θ by *induction* on the structure of its elements. For instance, if A is a set and we have functions $h_i : C_i(A) \to A$ for $1 \leq i \leq n$, we may define a function $f : \Theta \to A$ by

$$f(\theta) = h_i(C_i(f)(z)) \quad \text{whenever } \theta = c_i(z).$$

Some examples of free structures:

1. in the degenerate case $n = 0$, the sum of zero functors is the constant functor 0 yielding the empty set. Since $\emptyset$ is the initial object of **Set**, $\mathbf{Set}_0 \cong \mathbf{Set}$, so $(\emptyset, id_\emptyset)$ is the initial object of $\mathbf{Set}_0$ and the empty set is therefore the free structure on no constructors.

2. Let $n = 2$ and let C_1, C_2 be constant functors yielding two sets T, U, respectively. It is immediate that $(T + U, \iota_1, \iota_2)$ is the free structure on C_1, C_2, with ι_1, ι_2 the canonical injections. In the special case in which $T = U = 1 = \{*\}$, we get the set of Boolean values, and ι_1, ι_2 may be referred to as "true" and "false".

3. Let $n = 1$ and C_1 be the constant functor yielding a set K. In that case, it is easy to see that the free structure is K itself, with the constructor being the identity id_K. So every set is the free structure on... itself.

4. Let $n = 2$, suppose that C_1 is the constant functor yielding 1 and that C_2 is the identity functor. Then, if Θ is the underlying set of the free structure on C_1, C_2, it must come with two constructors

$$\begin{aligned} c_1 &: 1 \to \Theta, \\ c_2 &: \Theta \to \Theta, \end{aligned}$$

   such that $(\Theta, [c_1, c_2])$ is initial for the functor sending a set X to $1 + X$ (*i.e.*, which adds an element $* \notin X$ to X). We invite the reader to check that $(\mathbb{N}, 0, \mathrm{succ})$ is the free structure in this case. The function $[0, \mathrm{succ}] : 1 + \mathbb{N} \to \mathbb{N}$ sends $*$ to 0 and a natural number n to $n + 1$.

5. Let $n = 2$, $C_1 = 1$ and $C_2(X) = T \times X$ for a fixed set T. The free structure is $(T^*, \varepsilon, \mathrm{cons})$, where T^* is the set of finite lists (words) of elements of T, ε is the empty list and $\mathrm{cons} : T \times T^* \to T^*$ is the function adding an element to the head of a list. Of course, by letting $T = 1$ we obtain again the integers.

6. Some free structures may be specified in more than one way. For example, setting $T = 2$ (a set with two elements) in the above example, or choosing $n = 3$, $C_1 = 1$ and both C_2, C_3 equal to the identity functor are two different ways of defining binary words.

7. Let $n = 2$, $C_1 = 1$ and $C_2(X) = X \times X$. In this case, we get binary trees, with the one-root-one-leaf tree as the base case and the function building a tree from two trees as the other constructor.

8. Let $n = 2$, $C_1 = 1$ and $C_2(X) = T \to X$, *i.e.*, the set of functions from a fixed set T to X. In this case, we get the set of trees with branches indexed by T; the constructors are the one-root-one-leaf tree and the map building a tree from a T-indexed family of trees. Of course, if we let $T = 2$ we retrieve the above example.

The reader will have understood by now that any inductive date type, *i.e.*, a set of objects whose elements are defined by induction using a finite number of constructors, may be defined as a free structure.

4.2 Representing free structures in system F

In system **F**, we may imagine that types represent sets: X is a "generic" set, $T \to U$ is the function space, and we have seen that we may define the Cartesian product $T \times U$ and the disjoint sum $T + U$. Then, we may suppose that, in certain cases, the functors $C_1, \ldots, C_n$ may be represented by types of **F** containing a fixed free atom X representing the argument of the functor. The fact that the C_i are covariant will be reflected by the fact X may only occur positively.[2] For example:

- if C_1 and C_2 are as in example 4, and if we admit to have a type 1 representing the singleton set (which we actually do, as we shall see), we have $C_1 = 1$ and $C_2 = X$;
- in example 5, we have $C_1 = 1$ and $C_2 = T \times X$;
- in example 7, we have $C_1 = 1$ and $C_2 = X \times X$;
- in example 8, we have $C_1 = 1$ and $C_2 = T \to X$.

In such cases, the free structure on $C_1, \ldots, C_n$ may be represented in system **F** by the type

$$\Theta = \Pi X.(C_1 \to X) \to \cdots \to (C_1 \to X) \to X.$$

In fact, the constructors of Θ may be represented by terms of system **F**, as follows.

In the sequel, we write $x^T \vdash M : U$ to say that M is a system **F** term of type U containing a distinguished free variable x^T (of type T). Given a type C in which X occurs positively only, a type C' in which X occurs negatively only and a term $x^T \vdash M : U$, we build terms

$$\begin{aligned} x^{C[T/X]} &\vdash C(\!|M|\!) : C[U/X], \\ x^{C'[U/X]} &\vdash C(\!|M|\!) : C'[T/X], \end{aligned}$$

[2] An occurrence of X is *positive* (resp. *negative*) in C_i if it appears to the left of an even (resp. odd) number of arrows.

by induction on C and C':

- if C is an atom, there are two cases:
  - $C = X$, in which case $C(\!|M|\!) = M$;
  - $C = Y$, in which case $C(\!|M|\!) = x^Y$, independently of M.

  If C' is atom, only the second case can apply.

- if $C = D' \to E$, then observe that X must occur negatively only in D' and positively only in E. This means that we know how to inductively define

$$\begin{array}{rcl} x^{E[T/X]} & \vdash & E(\!|M|\!) : E[U/X], \\ x^{D'[U/X]} & \vdash & D'(\!|M|\!) : D'[T/X], \end{array}$$

  from which we set

$$C(\!|M|\!) = \lambda y^{D'[U/X]}.E(\!|M|\!)[x^{C[T/X]}D'(\!|M|\!)[y/x]/x].$$

  If $C' = D \to E'$, the definition is symmetric: just replace D' with D and E with E'.

- if $C = \Pi Y.D$, then X occurs only positively in D hand we know how to inductively define

$$x^{D[T/X]} \vdash D(\!|M|\!) : D[U/X].$$

  From this, we set

$$C(\!|M|\!) = \Lambda Y.D(\!|M|\!)[x^{\Pi Y.D[T/X]}Y/x].$$

  If $C' = \Pi Y.D'$, the situation is again similar: just replace D with D' and T with U.

Let now $C_1, \ldots, C_n$ be types containing X positively, as above. For $1 \leq i \leq n$, we set

$$M_i = x^\Theta X s_1^{C_1 \to X} \cdots s_n^{C_n \to X},$$

which is of type X. Using our notation, we have $x^\Theta \vdash M_i : X$, so we may build

$$x^{C_i[\Theta/X]} \vdash C_i(\!|M|\!) : C_i,$$

for all $1 \leq i \leq n$. From these, we define the terms

$$c_i = \lambda x^{C_i[\Theta/X]}.\Lambda X.\lambda s_1^{C_1 \to X} \ldots s_n^{C_n \to X}.s_i C_i(\!|M_i|\!),$$

which are of type $C_i[\Theta/X] \to \Theta$ and represent the constructor c_i of the free structure Θ.

As evidence of the fact that we have actually found a representation of the free structure in question, we give a general construction for inductively defining functions on the type Θ. Suppose we are given terms $H_i : C_i[T/X] \to T$ for all $1 \leq i \leq n$ and for some type T. We would like to define a term $x^\Theta \vdash F : T$ such that, for all $N : C_i[\Theta/X]$,

$$(\lambda x^\Theta.F)(c_i N) \to^* H_i(C_i(\!|F|\!)[N/x]),$$

which is what we expect if $\lambda x^{\Theta}.F$ has to represent the function of type $\Theta \to T$ defined by induction on Θ from $H_1, \ldots, H_n$. It is easy to verify that the solution is given by setting

$$F = x^{\Theta} T H_1 \cdots H_n.$$

Let us apply the construction to the examples given in the previous section:

1. if $n = 0$, then $\Theta = \Pi X.X$, which is the empty type, corresponding to absurdity ($\forall X.X$) and representing the empty set.

2. If $C_1 = T$ and $C_2 = U$, we have $\Theta = \Pi X.(T \to X) \to (U \to X) \to X$, corresponding to the polymorphic/second order encoding of the sum type/ disjunction.

   The special case of $T = 1$ (we shall see what is the definition of 1 in a moment) deserves attention. Since 1 logically corresponds to truth and $1 \Rightarrow X$ is the same as X, we may simplify the type $\Pi X.(1 \to X) \to (1 \to X) \to X$ into

   $$\mathbf{B} = \Pi X.X \to X \to X,$$

   which is the type of Booleans in system **F**. The two normal terms of type **B** are $\Lambda X.\lambda x^X y^X.x$ and $\Lambda X.\lambda x^X y^X.y$, representing true and false, respectively.

3. Two special cases are of interest. First, let $K = 1$. By operating the same simplification as above, we obtain the type

   $$\mathbf{U} = \Pi X.X \to X,$$

   which is the unit type, corresponding to the singleton set 1. Its only normal term is $\Lambda X.\lambda x^X.x$.

   The second case is $K = T \times U$. Since $(T \times U) \to X$ may be considered to be isomorphic to $T \to U \to X$, we get the type

   $$\Theta = \Pi X.(T \to U \to X) \to X,$$

   which is exactly the polymorphic/second order encoding of the product type/conjunction.

4. This case will be covered in detail in the next section.

5. If $C_1 = 1$ and $C_2 = T \times X$, with the simplifications $1 \to X = X$ and $(T \times X) \to X = T \to X \to X$, we get

   $$\Pi X.X \to (T \to X \to X) \to X,$$

   which is the type of lists of elements of type T.

6. If C_1 is as above and $C_2 = X \times X$, with the same simplifications we get the type of binary trees:

   $$\Pi X.X \to (X \to X \to X) \to X.$$

7. Similarly, if C_1 is as above and $C_2 = T \to X$, we get the type of tree of branching type T:

   $$\Pi X.X \to ((T \to X) \to X) \to X.$$

The last three cases are explained in detail in [GLT89]. Here, we shall content ourselves with giving the details of case 4, which is the most computationally relevant.

4.3 The case of integers: representable functions

If we apply the above construction, with the necessary optimizations, to the case of the two functor-types $C_1 = 1$ and $C_2 = X$, we obtain the type

$$\Pi X.X \to (X \to X) \to X.$$

Of course, the order of C_1, C_2 (and of $C_1, \ldots, C_n$ in general) is arbitrary, so we may as well consider the type

$$\mathbf{N} = \Pi X.(X \to X) \to X \to X,$$

which is what we shall take as the type of natural numbers in system $\mathbf{F}$.

The constructors obtained by pedantically applying the general definitions (but exchanging the order of C_1 and C_2 in the abstractions, and considering the optimization $1 \to X = X$) are

$$\begin{aligned} c_1 &= \lambda x^1.\Lambda X.\lambda s^{X\to X} z^X.z*, \\ c_2 &= \lambda x^{\mathbf{N}}.\Lambda X.\lambda s^{X\to X} z^X.s(xXsz). \end{aligned}$$

In c_1, the type 1 and the term $*$ (the only element of type 1) are purely formal; we may remove them in accordance with our optimization $1 \to X = X$ and obtain

$$\underline{0} = \Lambda X.\lambda s^{X\to X} z^X.z.$$

If we write $\mathrm{succ} = c_2$, we invite the reader to check that

$$\overbrace{\mathrm{succ}(\ldots\mathrm{succ}}^{n}\ \underline{0}\ldots) \to^* \Lambda X.\lambda s^{X\to X} z^X.\overbrace{s(\ldots s}^{n}\ z\ldots),$$

which, if we forget the type annotations, is exactly the so-called *Church numeral* $\underline{n}$, one of the standard representations of the natural number n in the λ-calculus. One can prove that all normal terms of type $\mathbf{N}$, with the exception of $\Lambda X.\lambda s^{X\to X}.s$, are of this form.[3] So the constructor $\underline{0}$ is nothing but the representation of 0, and the constructor succ represents the successor function, as expected from a representation of the natural numbers as a free structure.

The general induction scheme gives us, for any type T, $G : T$ and $H : T \to T$, the function

$$F = \lambda x^{\mathbf{N}}.xTHG : \mathbf{N} \to T,$$

which is immediately seen to represent the function $f(n) = h^n(g)$, if we consider T as a set, $g \in T$ is h a function on T.

From iteration, using polymorphism, we may obtain primitive recursive definitions. We recall that primitive recursion lets us define, from functions $g : \mathbb{N}^k \to \mathbb{N}$ and $h : \mathbb{N}^{n+2} \to \mathbb{N}$, the function $f : \mathbb{N}^{k+1} \to \mathbb{N}$ by

$$f(x, \vec{y}) = \begin{cases} g(\vec{y}) & \text{if } x = 0, \\ h(x-1, \vec{y}, f(x-1, \vec{y})) & \text{if } x > 0. \end{cases}$$

[3] See [GLT89], Sect. 15.1.1 for a proof.

So suppose we have

$$\begin{aligned} G &: \overbrace{\mathbf{N} \to \cdots \to \mathbf{N}}^{n} \to \mathbf{N}, \\ H &: \mathbf{N} \to \overbrace{\mathbf{N} \to \cdots \to \mathbf{N}}^{n} \to \mathbf{N} \to \mathbf{N}, \end{aligned}$$

corresponding to the functions g and h, respectively. We shall use product types, which we have seen are definable in system **F**. With these, we define

$$\begin{aligned} &\widehat{G} = \langle G y_1^{\mathbf{N}} \dots y_n^{\mathbf{N}}, \underline{0} \rangle : \mathbf{N} \times \mathbf{N}, \\ &\widehat{H} = \lambda p^{\mathbf{N} \times \mathbf{N}}. \langle H(\pi_2^{\mathbf{N},\mathbf{N}} p) y_1^{\mathbf{N}} \dots y_n^{\mathbf{N}} (\pi_1^{\mathbf{N},\mathbf{N}} p), \mathrm{succ}(\pi_2^{\mathbf{N},\mathbf{N}} p) \rangle : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \times \mathbf{N}, \end{aligned}$$

and then we set

$$F = \lambda x^{\mathbf{N}} y_1^{\mathbf{N}} \dots y_n^{\mathbf{N}}. \pi_1^{\mathbf{N},\mathbf{N}} (x(\mathbf{N} \times \mathbf{N}) \widehat{H} \widehat{G}) : \mathbf{N} \to \overbrace{\mathbf{N} \to \cdots \to \mathbf{N}}^{n} \to \mathbf{N}.$$

We let the reader check that F represents indeed the function f as defined above. The idea is that the computation is done on a pair of integers (m, n), with n being the current stage of the recursion and m being the partial result at the n-th stage. Computation starts with the pair $(g(\vec{y}), 0)$, and ends by projecting on the first component of the pair. If the argument leading the recursion is 0, we immediately obtain the desired result. Otherwise, one application of the term $\widehat{H}$ gives us the pair $(h(0, \vec{y}, g(\vec{y})), 1)$. If the argument leading the recursion is 1, we can stop and obtain the expected result. Otherwise, we reapply $\widehat{H}$, which gives us $(h(1, \vec{y}, h(0, \vec{y}, g(\vec{y})), 2)$, and so on.

Now that we know how to represent integers, we may give the following definition:

Definition 3 *A partial function $f : \mathbb{N} \to \mathbb{N}$ is* representable *in system* **F** *if there is a term $F : \mathbf{N} \to \mathbf{N}$ such that, for every $x \in \mathbb{N}$, if x is in the domain of f, we have*

$$M\underline{x} \to^* \underline{f(x)},$$

and if x is not in the domain of f, $M\underline{x}$ must have no normal form.

We have seen so far that in system **F** we can represent at least all primitive recursive functions. In fact, we can do much, much more. However, there are limits, the biggest one being that we may only represent *total* functions. In other words, the second clause of Definition 3 never applies. This is because, as we shall see, *every term of system* **F** *is normalizable.*

4.4 Towards normalization: the reducibility technique

A *normalization theorem* is a result stating that every object of a rewriting system has a normal form. There is a very simple and elegant proof of the normalization theorem for the simply typed λ-calculus, due to Tait [Tai67]. By the Curry-Howard isomorphism, and by the correspondence between normalization in natural deduction and cut-elimination in sequent calculus, this is actually an alternative proof of the cut-elimination theorem for propositional **LJ**. For simplicity, we shall restrict to the implication fragment only (so to **NM** and **LM**).

In the following, we call *neutral* a term which does not start with an abstraction. The fundamental property of neutral terms is that they are stable under application: if N is neutral, then for every M the term NM is still neutral.

For each simple type T (which, we remind, is either an atom X or of the form $U \to V$), we define:

- $\mathrm{Norm}(T)$ as the set of all normalizable terms of type T;
- $\mathrm{Neut}(T)$ as the set of all normal neutral terms of type T.

We also define the set of *reducible* terms of type T, denoted by $\mathrm{Red}(T)$, by induction on T, as follows:

- $\mathrm{Red}(X) = \mathrm{Norm}(X)$;
- $\mathrm{Red}(U \to V) = \{M : U \to V \mid \forall N \in \mathrm{Red}(U), MN \in \mathrm{Red}(V)\}$.

The normalization theorem is a consequence of the Adequacy and Adaptation Lemmas, given below. Each of them requires one auxiliary result.

Lemma 7 *For every type T, $\mathrm{Red}(T)$ is closed under β-expansion, i.e., if $M \in \mathrm{Red}(T)$ and $M_1 \to M$, then $M_1 \in \mathrm{Red}(T)$.*

Proof. By induction on T. For the atomic case, obviously normalizable terms are closed under β-expansion. So let $T = U \to V$, let $M \in \mathrm{Red}(U \to V)$ and let $M_1 \to M$. Take any $N \in \mathrm{Red}(U)$. To conclude, it is enough to show that $M_1 N \in \mathrm{Red}(V)$. But $M_1 N \to MN \in \mathrm{Red}(V)$, so we conclude by the induction hypothesis. □

Lemma 8 (Adequacy) *Let $M : U$ and let $x_1^{T_1}, \ldots, x_n^{T_n}$ be the free variables of M. For all $N_1 \in \mathrm{Red}(T_1), \ldots, N_n \in \mathrm{Red}(T_n)$, we have*

$$M[N_1/x_1, \ldots, N_n/x_n] \in \mathrm{Red}(U).$$

Proof. By induction on M. We shall use the shorthand notation $P[\vec{N}/\vec{x}]$ for denoting the simultaneous substitution of every N_i to x_i in a term P.

If $M = x^U$, the result is trivial.

Let $M = \lambda y^V.M'$, with $U = V \to V'$ and $M' : V'$. Let $P \in \mathrm{Red}(V)$. The induction hypothesis gives us $M_1 = M'[\vec{N}/\vec{x}][P/y] \in \mathrm{Red}(V')$. But $M[\vec{N}/\vec{x}]P \to M_1$, so we conclude by applying Lemma 7.

Let $M = PQ$, with $P : V \to U$ and $Q : V$. If we set $P' = P[\vec{N}/\vec{x}]$, $Q' = Q[\vec{N}/\vec{x}]$, using the induction hypothesis we have $P' \in \mathrm{Red}(V \to U)$, $Q' \in \mathrm{Red}(V)$. Then, by definition $M[\vec{N}/\vec{x}] = P'Q' \in \mathrm{Red}(U)$. □

The following is a simple property of λ-terms, which holds regardless of types (and we therefore formulate it without them).

Lemma 9 *For every λ-term M, if Mx is normalizable, then so is M.*

Proof. Consider a reduction $Mx \to^* N$, where N is a normal form. If all reducts of M (including M itself) are neutral, then we actually have $N = N'x$, with N' normal and neutral, and $M \to^* N'$, so we conclude. Otherwise, we must have $Mx \to^* (\lambda x.P)x \to P \to^* N$, which means that $M \to^* \lambda x.P \to^* \lambda x.N$, which is normal because N is. □

Lemma 10 (Adaptation) *For every type T,* $\mathrm{Neut}(T) \subseteq \mathrm{Red}(T) \subseteq \mathrm{Norm}(T)$.

Proof. Again, by induction on T. The atomic case is immediate from the definition. Let $T = U \to V$, and let $N : U \to V$ be normal and neutral. Take $M \in \mathrm{Red}(U)$. It is enough to show that $NM \in \mathrm{Red}(V)$. By definition, NM is still neutral. By the induction hypothesis, M is normalizable. If M_0 is its normal form, we have (using once more the induction hypothesis) $NM_0 \in \mathrm{Norm}(V) \subseteq \mathrm{Red}(V)$, so we conclude by applying Lemma 7.

Let now $M \in \mathrm{Red}(U \to V)$. We want to show that M is normalizable. Every variable is obviously normal and neutral, so by the induction hypothesis $x^U \in \mathrm{Red}(U)$. But by definition, $Mx^U \in \mathrm{Red}(V)$, so by the induction hypothesis it is normalizable and Lemma 9 allows us to conclude. □

Theorem 11 (Normalization for simple types) *Every simply-typed term is normalizable.*

Proof. Let $M : U$, with free variables $x_1^{T_1}, \ldots, x_n^{T_n}$. By the first inclusion of the Adaptation Lemma 10, for all $1 \leq i \leq n$, $x_i^{T_i} \in \mathrm{Red}(T_i)$. Then, the Adequacy Lemma 8 gives us that $M[x_1^{T_1}/x_1, \ldots, x_n^{T_n}/x_n] = M \in \mathrm{Red}(U)$, so we conclude by using the second inclusion of the Adaptation Lemma 10. □

4.5 Reducibility and polymorphism

The reducibility technique is very powerful. After minor adjustments, it can be used to prove the normalization theorem of Gödel's system **T**, which is not provable in first-order Peano arithmetic [GLT89]. It may also be adapted to prove the *strong normalization* theorem, that is to say that *every* reduction sequence eventually terminates (while normalization simply states that there is always *at least one* terminating sequence).

However, to make it work for system **F**, non-trivial modifications are needed. We immediately realize this by attempting to define $\mathrm{Red}(\Pi X.U)$. The intuition would be to set

$$\mathrm{Red}(\Pi X.U) = \{M : \Pi X.U \mid \forall V,\ MV \in \mathrm{Red}(U[V/X])\},$$

where V should range over *every* type, because any type may be substituted to X in $\Pi X.U$ (this is the meaning, and power, of polymorphism!). But then the definition is obviously ill-founded, because $U[V/X]$ may be much more complex than $\Pi X.U$ (think of $V = \Pi X.U$ itself...).

Girard solved this apparently inextricable tangle, which is due to the intrinsic impredicativity of second order quantification, by means of his so-called *reducibility candidates*. Instead of directly defining reducibility of a given type, we give an abstract notion of what a set of reducible terms should look like (a "candidate"), independently of types (and therefore of induction). Then, we define a *parametric* reducibility, in terms of a *valuation* which assigns a reducibility candidate to every free atom of the type under consideration. In the second-order case, instead of letting the type vary over all possible instances (which is incompatible with induction), we shall let the valuation vary over all possible assignments, which is sound because valuations are not defined by induction.

The key is to identify which abstract properties characterize reducibility. Originally, Girard found the right conditions by trial and error [GLT89]. Here,

we are able to identify them by inspecting the proof of Theorem 11: the only two properties we ever need to show about the sets $\mathrm{Red}(T)$ is that they are closed under β-expansion (Lemma 7) and that they enjoy the adaptation conditions (Lemma 10). The other lemmas and parts of the proof, although of course necessary, are not properties of the sets $\mathrm{Red}(T)$.

4.6 Normalization of system F

We start by extending the definition of neutrality:

Definition 4 (Neutrality for system F) *A term is neutral if it is neither an abstraction nor a universal abstraction.*

The sets $\mathrm{Norm}(T)$ and $\mathrm{Neut}(T)$ are defined just as in the simply-typed λ-calculus (with the updated notion of neutrality, of course).

Definition 5 (Reducibility candidate) *A* reducibility candidate *of type T is a set R of terms of type T such that:*

CR1. *R is closed under β-expansion;*

CR2. $\mathrm{Neut}(T) \subseteq R \subseteq \mathrm{Norm}(T)$.

We denote by $\mathrm{CR}(T)$ the set of all reducibility candidates of type T.

For any type T, the set $\mathrm{Norm}(T)$ is the prototypical reducibility candidate of type T. Therefore, $\mathrm{CR}(T)$ is never empty.

Definition 6 (Valuation) *A* valuation *is a partial function ρ from atomic types to reducibility candidates, whose domain is finite and denoted by $\mathrm{dom}\rho$.*

If R is a reducibility candidate, we denote by $\rho[X := R]$ the valuation whose domain is $\mathrm{dom}\rho \cup \{X\}$ and which coincides with ρ everywhere except on X, where it is equal to R.

A valuation ρ covers *a type T if the free atoms of T are all in $\mathrm{dom}\rho$; it* covers *a term $M : U$ whose free variables are $x_1^{T_1}, \ldots, x_n^{T_n}$ if it covers every one of $U, T_1, \ldots, T_n$.*

The default valuation *on $\Gamma = X_1, \ldots, X_n$ is the valuation ρ_Γ such that $\mathrm{dom}\rho_\Gamma = \{X_1, \ldots, X_n\}$ and such that $\rho_\Gamma(X_i) = \mathrm{Norm}(X_i)$ for all $1 \leq i \leq n$.*

Definition 7 (Substitution induced by a valuation) *Let ρ be a valuation, and let X be an atomic type. We set*

$$X^\rho = \begin{cases} U & \text{if } X \in \mathrm{dom}\rho \text{ and } U \text{ is the type of } \rho(X); \\ X & \text{if } X \notin \mathrm{dom}\rho. \end{cases}$$

Let T be a type whose free atoms are $X_1, \ldots, X_n$. We set

$$T^\rho = T[X_1^\rho / X_1, \ldots, X_n^\rho / X_n].$$

Similarly, if $M : T$ and $X_1, \ldots, X_n$ are all the free atoms appearing in T and in the types of the free variables of M, we define

$$M^\rho = M[X_1^\rho / X_1, \ldots, X_n^\rho / X_n],$$

which is of type T^ρ.

Observe that, if $M : T$ and if Γ contains all the free atoms appearing in M and T, then $T^{\rho_\Gamma} = T$ and $M^{\rho_\Gamma} = M$, *i.e.*, the default valuation induces the identity substitution.

Given a type T and a valuation ρ covering T, we define the set of terms $\mathrm{Red}_\rho(T)$ by induction on T:

- $\mathrm{Red}_\rho(X) = \rho(X)$;
- $\mathrm{Red}_\rho(U \to V) = \{M : (U \to V)^\rho \mid \forall N \in \mathrm{Red}_\rho(U),\ MN \in \mathrm{Red}_\rho(V)\}$;
- $\mathrm{Red}_\rho(\Pi X.U) = \{M : (\Pi X.U)^\rho \mid \forall V\, \forall R \in \mathrm{CR}(V),\ MV \in \mathrm{Red}_{\rho[X:=R]}(U)\}$.

Lemma 12 *Let $M : \Pi X.U$, let V be a type, and suppose that MV is normalizable. Then, M is normalizable.*

Proof. Entirely similar to the proof of Lemma 9. □

Lemma 13 *For every type T and valuation ρ covering T, $\mathrm{Red}_\rho(T) \in \mathrm{CR}(T^\rho)$.*

Proof. The fact that all terms in $\mathrm{Red}_\rho(T)$ have type T^ρ is immediate from the definition. Properties CR1 and CR2 are verified by induction on T.

The atomic case is trivial. For the arrow case, we follow exactly the arguments given for Lemma 7 (for CR1) and Lemma 10 (for CR2). So we may assume that $T = \Pi X.U$.

For CR1, suppose that $M_1 \to M \in \mathrm{Red}_\rho(T)$. Take any type V and any $R \in \mathrm{CR}(V)$. We have $M_1 V \to MV \in \mathrm{Red}_{\rho[X:=R]}(U)$ by hypothesis, so we infer $M_1 V \in \mathrm{Red}_{\rho[X:=R]}(U)$ by applying CR1, which holds for U. But, by genericity of V and R, this is enough to conclude that $M_1 \in \mathrm{Red}_\rho(T)$.

For what concerns the first inclusion of CR2, let $N \in \mathrm{Neut}(T^\rho)$. Take any type V and any $R \in \mathrm{CR}(V)$. Since N is neutral and normal, NV is also neutral and normal, of type $U^{\rho[X:=R]}$. But CR2 holds for U, so $NV \in \mathrm{Red}_{\rho[X:=R]}(U)$, which is enough to show that $N \in \mathrm{Red}_\rho(T)$.

Let us turn to the second inclusion of CR2. Let $M \in \mathrm{Red}_\rho(T)$, let V be any type and $R \in \mathrm{CR}(V)$. Since CR2 holds for U, we know that $MV \in \mathrm{Red}_{\rho[X:=R]}(U) \subseteq \mathrm{Norm}(U^{\rho[X:=R]})$. But then Lemma 12 allows us to conclude that $M \in \mathrm{Norm}(T^\rho)$. □

Lemma 14 (Substitution) *For every type V, W and every assignment ρ covering them, we have $\mathrm{Red}_\rho(V[W/Y]) = \mathrm{Red}_{\rho[Y:=\mathrm{Red}_\rho(W)]}(V)$.*

Proof. A straightforward induction on V. □

Lemma 15 (Adequacy) *Let $M : U$ and let $x_1^{T_1}, \ldots, x_n^{T_n}$ be the free variables of M. For every valuation ρ covering M and for every $N_1 \in \mathrm{Red}_\rho(T_1), \ldots, N_n \in \mathrm{Red}_\rho(T_n)$, we have*

$$M^\rho[N_1/x_1, \ldots, N_n/x_n] \in \mathrm{Red}_\rho(U).$$

PROOF. By induction on M. We shall use the shorthand notation $P[\vec{N}/\vec{x}]$ for denoting the simultaneous substitution of every N_i to x_i in a term P.

If $M = x^U$, the result is trivial.

Let $M = \lambda y^V.M_1$, with $U = V \to V_1$ and $M_1 : V_1$. Let $P \in \mathrm{Red}_\rho(V)$. The induction hypothesis gives us $M_2 = M_1^\rho[\vec{N}/\vec{x}][P/y] \in \mathrm{Red}_\rho(V_1)$. But $M^\rho[\vec{N}/\vec{x}]P \to M_2$, so we conclude by CR1 (which holds thanks to Lemma 13).

Let $M = PQ$, with $P : V \to U$ and $Q : V$. If we set $P_1 = P^\rho[\vec{N}/\vec{x}]$, $Q_1 = Q^\rho[\vec{N}/\vec{x}]$, using the induction hypothesis we have $P_1 \in \mathrm{Red}_\rho(V \to U)$, $Q_1 \in \mathrm{Red}_\rho(V)$. Then, by definition $M^\rho[\vec{N}/\vec{x}] = P_1Q_1 \in \mathrm{Red}_\rho(U)$.

Let $M = \Lambda Y.M_1$, with $U = \Pi Y.U_1$ and $M_1 : U_1$. Note that, by α-equivalence, we may always suppose $Y \notin \mathrm{dom}\rho$. Let V be any type, let $R \in \mathrm{CR}(V)$, and let $M_2 = M_1^{\rho[Y:=R]}[\vec{N}/\vec{X}]$. The induction hypothesis gives us $M_2 \in \mathrm{Red}_{\rho[Y:=R]}(U_1)$. But by definition $M_1^{\rho[Y:=R]} = M_1^\rho[V/Y]$, so $M^\rho[\vec{N}/\vec{x}]V \to M_2$, which is enough to prove $M^\rho \in \mathrm{Red}_\rho(U)$.

Let $M = PW$ with $P : \Pi Y.V$ and W a type, so that $U = V[W/Y]$. Let $P_1 = P^\rho[\vec{N}/\vec{X}]$. By induction, we know that $P_1 \in \mathrm{Red}_\rho(\Pi Y.V)$, which gives us that $P_1W^\rho \in \mathrm{Red}_{\rho[Y:=\mathrm{Red}_\rho(W)]}(V)$. But $P_1W^\rho = M^\rho[\vec{N}/\vec{x}]$, so we conclude by applying Lemma 14. □

Theorem 16 (Normalization of system F) *Every term of system* **F** *is normalizable.*

PROOF. Let $M : U$ be a system **F** term, whose free variables are $x_1^{T_1}, \ldots, x_n^{T_n}$ and such that the free atoms of $U, T_1, \ldots, T_n$ are in the list Γ. By the first inclusion of CR2, we have $x_i^{T_i} \in \mathrm{Red}_{\rho_\Gamma}(T_i)$ (remember that $T_i^{\rho_\Gamma} = T_i$). Hence, by the Adequacy Lemma 15, we have $M^{\rho_\Gamma}[x_1^{T_1}/x_1, \ldots, x_n^{T_n}/x_n] = M \in \mathrm{Red}_{\rho_\Gamma}(U)$, so we conclude by the second inclusion of CR2. □

5 Second-order arithmetic

5.1 Second-order Peano arithmetic

Second-order Peano arithmetic is the original formal system introduced by Peano to formalize number theory. The fact that it is a full second-order theory, *i.e.*, that it has first-order *and* second-order quantification, allows the language to quantify not only over *individuals* of the domain of discourse (*i.e.*, the integers), but also on *properties* of such individuals (*i.e.*, sets of integers).

The formulation we give below, which we call $\mathbf{PA_2}$, is due to Takeuti. It is proof-theoretically more amenable than Peano's original formulation, without serious differences in terms of expressiveness: the theory is enormously more expressive than its first-order counterpart. However, the categoricity of the theory is lost (by contrast, second-order Peano arithmetic is well known to be categorical, *i.e.*, to admit only one model modulo isomorphism).

The *terms* of the language of $\mathbf{PA_2}$ are given by

$$t, u, v ::= x \mid f(t_1, \ldots, t_n),$$

where x ranges over a denumerably infinite set of *first-order variables*, and f ranges over a denumerably infinite set of *functional symbols*, each coming with its *arity* (equal to n above). We suppose that the functional symbols of arity n are in bijection with the primitive recursive functions $\mathbb{N}^n \to \mathbb{N}$. Among these, there is a nullary symbol $\mathsf{0}$, representing the constant $0 \in \mathbb{N}$, and a unary symbol S, representing the successor.

The formulas of $\mathbf{PA_2}$ are given by

$$A, B ::= t = u \mid (t_1, \ldots, t_n) \in X \mid A \Rightarrow B \mid \forall x.A \mid \exists x.A \mid \forall X.A,$$

where X ranges over a denumerably infinite set of *second-order variables*, each coming with its *arity* (equal to n above). Observe that the symbol '$\in$' appearing in the atomic formula $(t_1, \ldots, t_n) \in X$ (which we may abbreviate by $\vec{t} \in X$) is *not* part of the language, *i.e.*, there is no membership relation in the language. We only use the symbol as a convenient (and, we believe, more readable) notation for the more usual $X\vec{t}$. In case the arity of the second-order variable is zero, we just write X instead of $() \in X$.

Additional connectives (negation, conjunction, second order existential quantification, etc.) are defined from second-order quantification. As usual, $t \neq u$ is short for $\neg(t = u)$. Note that first-order existential quantification may also be encoded by $\exists x.A := \forall X.(\forall x.A \Rightarrow X) \Rightarrow X$, but we choose to include it in the language for the sake of the exposition.

The rules of $\mathbf{PA_2}$ are those of first-order **NK** plus the rules for second order quantification, of which introduction is formulated as usual, while elimination is formulated as follows:

$$\frac{\forall X.A}{A[B/Xx_1 \ldots x_n]}$$

where the notation $A[B/Xx_1 \ldots x_n]$ means that every occurrence of the atom $(t_1, \ldots, t_n) \in X$ in A, with X free, is replaced by the formula $B[t_1/x_1, \ldots, t_n/x_n]$. For instance, $(\mathsf{0} \in X)[\mathsf{S0} = x/Xx]$ is equal to $\mathsf{S0} = \mathsf{0}$.

Finally, $\mathbf{PA_2}$ has the following axioms (the notations are explained below):

$$\begin{array}{rcl} E_r & := & \forall x.x = x, \\ E_L & := & \forall xy.x = y \Rightarrow x =_L y, \\ E_{pr} & := & \text{universally closed equations defining primitive recursive functions,} \\ P_4 & := & \forall x.\mathsf{S}x \neq \mathsf{0}, \\ I & := & \forall x.\mathsf{Int}(x). \end{array}$$

Axiom E_r states the reflexivity of equality. In the axiom E_L, we used the definition

$$x =_L y \quad := \quad \forall Z.x \in Z \Rightarrow y \in Z,$$

which is known as *Leibniz equality*, so the axiom says that equality implies Leibniz equality. Using the reflexivity axiom and a second-order elimination rule with the formula $x = z$, we may prove the converse:

$$\dfrac{\dfrac{x =_L y}{x = x \Rightarrow x = y} \quad \dfrac{E_r}{x = x}}{x = y}$$

Therefore, axioms E_r and E_L together state that equality and Leibniz equality coincide. This fact may be used to derive all of the other fundamental properties of equality, namely symmetry, transitivity, and contextuality:

$$\begin{array}{rcl} E_s & := & \forall xy.x = y \Rightarrow y = x \\ E_t & := & \forall xyz.x = y \Rightarrow y = z \Rightarrow x = z \\ E_u & := & \forall xy.x = y \Rightarrow u[x/z] = u[y/z] \quad \text{for every term } u \end{array}$$

The axiom E_L may also be used to derive the following rule, for any formula A:

$$\frac{t = u \quad A[t/x]}{A[u/x]}$$

Leibniz equality trivially satisfies also the reflexivity axiom, so one may eliminate altogether the equality symbol from the language and use Leibniz equality as the definition of equality, thus avoiding the axioms E_r and E_L. This is indeed a possible "optimization" of the formulation of $\mathbf{PA_2}$.[4] However, we chose to include a primitive equality relation for the sake of the exposition.

The infinitely many axioms of the form E_{pr} are needed to "use" the function symbols of the language: since each function symbol corresponds to a recursive function, each symbol comes with suitable equational axioms corresponding to the definition of that function. For instance, there will be a binary function symbol $+$ (for which we use infix notation) which comes with the axioms $\forall y.\mathsf{0} + y = y$ and $\forall xy.\mathsf{S}x + y = \mathsf{S}(x + y)$.

The axiom P_4 (traditionally referred to as Peano's 4th axiom), stating that zero is not the successor of any number, is the only axiom forcing an *inequality*. It rules out the possibility of trivial models, *i.e.*, models whose domain is a singleton. Combined with the injectivity of the successor, it implies that $\mathbf{PA_2}$ has no finite model. The injectivity of the successor, traditionally known as Peano's 3rd axiom, is the formula

$$P_3 := \forall xy.\mathsf{S}x = \mathsf{S}y \Rightarrow x = y,$$

which is derivable in our formulation of the theory, using the equation defining the predecessor function P (namely, $\forall z.\mathsf{P}(\mathsf{S}z) = z$), plus the axioms and rules concerning equality which we derived above:

$$\dfrac{\dfrac{\forall z.\mathsf{P}(\mathsf{S}z) = z}{\mathsf{P}(\mathsf{S}y) = y} \quad \dfrac{\dfrac{\forall z.\mathsf{P}(\mathsf{S}z) = z}{\mathsf{P}(\mathsf{S}x) = x} \quad \dfrac{\mathsf{S}x = \mathsf{S}y \Rightarrow \mathsf{P}(\mathsf{S}x) = \mathsf{P}(\mathsf{S}y) \quad [\mathsf{S}x = \mathsf{S}y]^*}{\mathsf{P}(\mathsf{S}x) = \mathsf{P}(\mathsf{S}y)} \; \vdots\, E_{\mathsf{P}z}}{x = \mathsf{P}(\mathsf{S}y)}}{x = y}}{\dfrac{\mathsf{S}x = \mathsf{S}y \Rightarrow x = y}{P_3}}\,{}^*$$

[4] One may "optimize" even further by suppressing all axioms except P_4 and I, and adding the above rule stating $A[t/x] \Rightarrow A[u/x]$ as soon as the equation $t = u$ is derivable from primitive recursive equations and the basic properties of equality (reflexivity, symmetry, transitivity, contextuality). This is the choice made by Krivine in his system AF_2 (*arithmétique fonctionnelle du second ordre* [Kri93]).

In the axiom I, we used the formula

$$\mathsf{Int}(x) := \forall X.\mathsf{0} \in X \Rightarrow (\forall y.y \in X \Rightarrow \mathsf{S}y \in X) \Rightarrow x \in X.$$

This axiom is the essence of Peano arithmetic: it is the principle of induction. Indeed, the formula $\mathsf{Int}(x)$ asserts that x is an integer: it belongs to any set X which contains zero and is closed under successor. Therefore, axiom I is stating "every individual is an integer", which means that we may freely apply induction to every term of the language.

Second order Peano arithmetic may be further endowed with an extra axiom providing some form of choice. For instance, Krivine [Kri04] calls "analysis" the system we refer to as $\mathbf{PA_2}$ plus the axiom of dependent choice. This is because dependent choice is enough for proving all standard results of analysis. In these notes we shall not consider any form of the axiom of choice.

5.2 Second-order Heyting arithmetic

Second-order *Heyting arithmetic* ($\mathbf{HA_2}$) has exactly the same language and axioms of $\mathbf{PA_2}$, but we no longer allow proving A from its double negation. In other words, $\mathbf{HA_2}$ is the *intuitionistic* version of second-order Peano arithmetic. The two systems are related by Gödel-Gentzen's negative translation, of which we shall present here a generalization, due to Friedman.

We write $A \vdash_{\mathrm{J}} B$ (resp. $A \vdash_{\mathrm{K}} B$) to mean that B is an intuitionistic (resp. classical) consequence of A, *i.e.*, B can be derived from A in **NJ** (resp. **NK**). Similarly, we write $A \dashv\vdash_{\mathrm{J}} B$ (resp. $A \dashv\vdash_{\mathrm{K}} B$) to mean that A and B are intuitionistically (resp. classically) equivalent. Then, the notations $\mathbf{PA_2} \vdash A$ and $\mathbf{HA_2} \vdash A$ will mean

$$E_r, E_L, E^1_{pr}, \dots, E^n_{pr}, P_4, I \vdash_{\mathrm{K}} A$$

and

$$E_r, E_L, E^1_{pr}, \dots, E^n_{pr}, P_4, I \vdash_{\mathrm{J}} A,$$

respectively, where $E^1_{pr}, \dots, E^n_{pr}$ is an arbitrary number of axioms expressing primitive recursive equations.

Le A, R be arithmetic formulas. We set $\neg_R A := A \Rightarrow R$, and define A^R by induction on A:

$$\begin{aligned} A^R &= \neg_R\neg_R A && \text{if } A \text{ is atomic} \\ (B \Rightarrow C)^R &= B^R \Rightarrow C^R \\ (\forall x.B)^R &= \forall x.B^R \\ (\exists x.B)^R &= \neg_R\neg_R\exists x.B^R \\ (\forall X.B)^R &= \forall X.B^R \end{aligned}$$

In other words A^R is obtained from A by replacing every atom C with $\neg_R\neg_R C$, and by prefixing every existential quantifier with a (relative) double negation.

In the following, R always denotes a generic formula.

Lemma 17 *Let A, B be formulas.*

1. $\bot^R \dashv\vdash_{\mathrm{J}} R$;

2. $\neg_R\neg_R\neg_R A \vdash_{\mathrm{J}} \neg_R A$ *and, therefore,* $\neg_R\neg_R\neg_R\neg_R A \vdash_{\mathrm{J}} \neg_R\neg_R A$;

3. $\neg_R\neg_R(A \Rightarrow B) \vdash_{\mathrm{J}} A \Rightarrow \neg_R\neg_R B$;

4. $\neg_R\neg_R\forall\xi.A \vdash_{\mathrm{J}} \forall\xi.\neg_R\neg_R A$, *where* ξ *is a variable of any order;*

5. $\neg_R\neg_R\exists x.A \vdash_{\mathrm{J}} \neg_R\forall x.\neg_R A$ *(this also holds for second order, but we will only need it for first order, in the proof of Friedman's Theorem 28).*

Proof. The proof consists of the following deductions:

1.
$$\dfrac{\dfrac{\perp^R = \forall X.(X \Rightarrow R) \Rightarrow R}{(R \Rightarrow R) \Rightarrow R} \quad \dfrac{[R]}{R \Rightarrow R}}{R} \qquad \dfrac{\dfrac{R}{(X \Rightarrow R) \Rightarrow R}}{\perp^R}$$

2.
$$\dfrac{\neg_R\neg_R\neg_R A \qquad \dfrac{\dfrac{[\neg_R A]^* \quad [A]^\dagger}{R}}{\neg_R\neg_R A}\ ^*}{\dfrac{R}{\neg_R A}\ ^\dagger}$$

3.
$$\dfrac{\neg_R\neg_R(A \Rightarrow B) \qquad \dfrac{\dfrac{[\neg_R B]^\dagger \quad \dfrac{[A \Rightarrow B]^* \quad [A]^\sharp}{B}}{R}}{\neg_R(A \Rightarrow B)}\ ^*}{\dfrac{\dfrac{R}{\neg_R\neg_R B}\ ^\dagger}{A \Rightarrow \neg_R\neg_R B}\ ^\sharp}$$

4.
$$\dfrac{\neg_R\neg_R\forall\xi.A \qquad \dfrac{\dfrac{[\neg_R A]^\dagger \quad \dfrac{[\forall\xi.A]^*}{A}}{R}}{\neg_R\forall\xi.A}\ ^*}{\dfrac{\dfrac{R}{\neg_R\neg_R A}\ ^\dagger}{\forall\xi.\neg_R\neg_R A}}$$

5.
$$\dfrac{\neg_R\neg_R\exists x.A \qquad \dfrac{\dfrac{[\exists x.A]^\dagger \qquad \dfrac{\dfrac{[\forall x.\neg_R A]^\ddagger}{\neg_R A} \quad [A]^*}{R}}{R}\ ^*}{\neg_R\exists x.A}\ ^\dagger}{\dfrac{R}{\neg_R\forall x.\neg_R A}\ ^\ddagger}$$

□

Lemma 18 *For every formula A, $\neg_R\neg_R A^R \vdash_{\mathrm{J}} A^R$.*

Proof. By induction on A, using Lemma 17. The atomic case is point 2; the implication case uses point 3; the universal quantification cases use point 4; and the existential quantification case uses again point 2. □

Note that the double negation introduced in front of existential quantifiers is fundamental to obtain Lemma 18: if we had defined $(\exists x.A)^R = \exists x.A^R$, we would be stuck, because $\neg_R\neg_R\exists x.A$ does not intuitionistically imply $\exists x.\neg_R\neg_R A$.

Lemma 19 *For every formula A, $(\neg\neg A)^R \vdash_{\mathrm{J}} A^R$.*

Proof. By definition, $(\neg\neg A)^R = (A^R \Rightarrow \bot^R) \Rightarrow \bot^R$. By point 1 of Lemma 17, this is intuitionistically equivalent to $\neg_R\neg_R A^R$, hence the result follows by Lemma 18. □

Lemma 20 *Let A be a formula. Then:*

1. *for every term t, $(A[t/x])^R = A^R[t/x]$;*
2. *for every formula B, $(A[B/X\vec{x}])^R \dashv\vdash_{\mathrm{J}} A^R[B^R/X\vec{x}]$.*

Proof. A straightforward induction on A. □

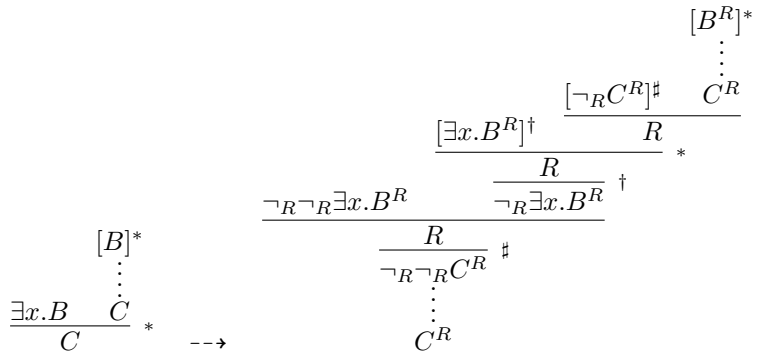
Theorem 21 *Let Γ be a sequence of formulas, let A be a formula, and let R be a formula whose free variables (of any order) are not quantified in Γ or A. Then, $\Gamma \vdash_{\mathrm{K}} A$ implies $\Gamma^R \vdash_{\mathrm{J}} A^R$.*

Proof. By induction on the last rule of the proof of $\Gamma \vdash_{\mathrm{K}} A$. The cases of an assumption and of the introduction or elimination of an implication are trivial. The case of a universal quantification uses Lemma 20 (point 1 for first-order, point 2 for second-order); the introduction case also uses the assumption that no free variable of R is quantified in Γ, A. The case of the introduction of an existential quantifier gives, using the induction hypothesis and point 1 of Lemma 20, the following derivation:

$$\dfrac{\dfrac{\vdots}{B[t/x]}}{\exists x.B} \quad \dashrightarrow \quad \dfrac{\dfrac{[\neg_R\exists x.B^R]^* \quad \dfrac{\vdots}{B^R[t/x]}}{R}}{\neg_R\neg_R\exists x.B^R}\ ^*$$

The case of the elimination of existential quantifier gives, using the induction

hypothesis, point 1 of Lemma 20, and Lemma 18,



The last case to be examined is a double-negation elimination rule. But this is precisely Lemma 19. □

Corollary 22 (Gödel, Gentzen) *There exists a transformation of formulas* $(\cdot)^*$ *such that, for every formula* A*:*

1. $A^* \dashv\vdash_{\mathrm{K}} A$*;*
2. $\vdash_{\mathrm{K}} A$ *iff* $\vdash_{\mathrm{J}} A^*$*.*

Proof. It is enough to let A^* be A^R with $R = \bot$. □

Corollary 23 (Gödel, Gentzen) *Classical logic and intuitionistic logic are equiconsistent.*

Proof. Obviously, if intuitionistic logic is inconsistent, then so is classical logic. For the converse, suppose $\vdash_{\mathrm{K}} \bot$. Then, $\vdash_{\mathrm{J}} \bot^*$, which we know by point 1 of Lemma 17 to be intuitionistically equivalent to $\bot$. □

The results of Corollary 22 tell us that it is not quite fair to think of intuitionistic logic simply as a restriction of classical logic. It is more correct to think of it as a *refinement*: intuitionistic logic is sensitive to distinctions which classical logic ignores (typically, the difference between existence and impossibility of non-existence) but is not really less powerful because, for every classically provable formula, intuitionistic logic proves a formula which is classically equivalent to it.

In the propositional case, a much simpler translation works:

Corollary 24 (Glivenko) *If* A *is propositional, then* $\vdash_{\mathrm{K}} A$ *iff* $\vdash_{\mathrm{J}} \neg\neg A$*.*

Proof. It is enough to show that, in the propositional case, $A^* \vdash_{\mathrm{J}} \neg\neg A$. Actually, we show that $A^* \dashv\vdash_{\mathrm{J}} \neg\neg A$. This is done by induction on A. The atomic case is trivial, because $A^* = \neg\neg A$. Let $A = B \Rightarrow C$. We shall first prove that $\neg\neg B \Rightarrow \neg\neg C \dashv\vdash_{\mathrm{J}} \neg\neg(B \Rightarrow C)$. The forward implication is shown by the

following derivation:

$$\dfrac{\dfrac{\neg\neg B \Rightarrow \neg\neg C \qquad \dfrac{\dfrac{[\neg(B \Rightarrow C)]^{\ddagger} \qquad \dfrac{\dfrac{[\neg B]^{\dagger} \quad [B]^{*}}{\bot}}{C}\,{}^{*}}{B \Rightarrow C}\,{}^{\dagger}}{\bot}}{\neg\neg B}\,{}^{\dagger}}{\neg\neg C} \qquad \dfrac{\dfrac{[\neg(B \Rightarrow C)]^{\ddagger} \quad \dfrac{[C]^{\sharp}}{B \Rightarrow C}}{\bot}}{\neg C}\,{}^{\sharp}}{\dfrac{\bot}{\neg\neg(B \Rightarrow C)}\,{}^{\ddagger}}$$

For the backward implication, we have:

$$\dfrac{\dfrac{[\neg\neg B]^{\ddagger} \qquad \dfrac{\dfrac{\neg\neg(B \Rightarrow C) \qquad \dfrac{\dfrac{[\neg C]^{\sharp} \qquad \dfrac{[B \Rightarrow C]^{*} \quad [B]^{\dagger}}{C}}{\bot}}{\neg(B \Rightarrow C)}\,{}^{*}}{\bot}}{\neg B}\,{}^{\dagger}}{\bot}}{\neg\neg C}\,{}^{\sharp}}{\neg\neg B \Rightarrow \neg\neg C}\,{}^{\ddagger}$$

Now, using the above and the induction hypothesis, we have $A^* = B^* \Rightarrow C^* \dashv\vdash_{\mathrm{J}} \neg\neg B \Rightarrow \neg\neg C \dashv\vdash_{\mathrm{J}} \neg\neg(B \Rightarrow C)$. □

Of course, Glivenko's theorem is false in presence of quantifiers, because the converse of point 5 of Lemma 17 is false, even when $R = \bot$. This is why a more complex translation is needed.

The translation also works for theorems of arithmetic:

Lemma 25 *For every R and for every axiom A of $\mathbf{PA_2}$, $A \vdash_{\mathrm{J}} A^R$.*

Proof. Given a closed formula of the form $\forall\vec{x}.t = u$, it is immediate to show that $\forall\vec{x}.t = u \vdash_{\mathrm{J}} (\forall\vec{x}.t = u)^R$. This covers the case of the axiom E_r and of all the primitive recursive axioms E_{pr}. For the axioms E_L and P_4, the proof

consists of the following derivations:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{E_L}{\overline{\overline{x = y \Rightarrow x =_L y}}} \quad [x = y]^{\#}}{x =_L y}}{x \in Z \Rightarrow y \in Z} \qquad [x \in Z]^{*}}{y \in Z}}{[\neg_R(y \in Z)]^{\$} \qquad \dfrac{R}{\neg_R(x \in Z)}\ {}^{*}}}{[(x \in Z)^R]^{\dagger} \qquad \dfrac{R}{\neg_R(x = y)}\ {}^{\#}}}{\dfrac{[(x = y)^R]^{\ddagger}}{\dfrac{R}{(y \in Z)^R}\ {}^{\$}}}}{\dfrac{(x \in Z)^R \Rightarrow (y \in Z)^R}{(x =_L y)^R}\ {}^{\dagger}}}{\dfrac{(x = y)^R \Rightarrow (x =_L y)^R}{\overline{\overline{E_L^R}}}\ {}^{\ddagger}}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P_4}{\mathsf{S}x \neq \mathsf{0}} \quad [\mathsf{S}x = \mathsf{0}]^{\dagger}}{\dfrac{\bot}{R}}}{[(\mathsf{S}x = \mathsf{0})^R]^{*} \qquad \dfrac{R}{\neg_R(\mathsf{S}x = \mathsf{0})}\ {}^{\dagger}}}{R}}{\vdots \text{ Point 1 of Lemma 17}}}{\dfrac{\dfrac{\bot^R}{(\mathsf{S}x = \mathsf{0})^R \Rightarrow \bot^R}\ {}^{*}}{P_4^R}}$$

Finally, the case of the axiom I is immediate: we obtain $\mathsf{Int}(x)$ by eliminating the first-order quantifier, then we prove $\mathsf{Int}(x)^R$ simply by instantiating the variable X with the formula $\neg_R\neg_R(z \in X)$ using a second-order elimination rule. From that, we re-introduce the first-order quantifier and get I^R. □

Corollary 26 (Gödel) $\mathbf{PA_2} \vdash A$ *iff* $\mathbf{HA_2} \vdash A^*$. *In particular, Peano arithmetic and Heyting arithmetic are equiconsistent.*

Proof. By Corollary 22, we know that $E_r, E_L, E^1_{pr}, \ldots, E^n_{pr}, P_4, I \vdash_{\mathrm{K}} A$ implies $E^*_r, E^*_L, (E^1_{pr})^*, \ldots, (E^n_{pr})^*, P^*_4, I^* \vdash_{\mathrm{J}} A^*$, so we apply Lemma 25 to conclude $\mathbf{HA_2} \vdash A^*$. To show equiconsistency, suppose that $\mathbf{PA_2}$ is inconsistent. Then, $\mathbf{PA_2} \vdash \mathsf{S0} = \mathsf{0}$, which gives us $\mathbf{HA_2} \vdash \neg\neg(\mathsf{S0} = \mathsf{0})$, which is enough to obtain a contradiction by modus ponens from $\neg(\mathsf{S0} = \mathsf{0})$ (an instance of P_4). □

The translation originally defined by Gödel (and, independently, by Gentzen) was the one we are denoting by $(\cdot)^*$, *i.e.*, it used "real" double negations. The generalization we presented here, which uses "parametric" double negations, was introduced by Friedman to show that that Peano arithmetic is conservative over Heyting arithmetic for "simple" formulas, as follows.

By assumption, the language of $\mathbf{PA_2}$ contains a binary function symbol leq which represents the function

$$\mathrm{leq}(x,y) = \begin{cases} 0 & \text{if } x \leq y, \\ 1 & \text{if } x > y. \end{cases}$$

From this, we define the formula

$$x \leq y \quad := \quad \mathrm{leq}(x,y) = 0,$$

which contains x and y free. Then, we define *bounded first-order quantifiers* as

$$\begin{aligned} \forall x \leq t.A &:= \forall x.x \leq t \Rightarrow A, \\ \exists x \leq t.A &:= \exists x.x \leq t \wedge A, \end{aligned}$$

where x is not free in t and we are using the second-order encoding of conjunction.

We define Δ_0^0 to be the set of formulas A such that

$$\mathbf{PA_2} \vdash A \Leftrightarrow \Phi,$$

where Φ is a second-order-closed formula belonging to the following fragment of $\mathbf{PA_2}$:

$$\Phi, \Psi \quad := \quad t = u \mid X \mid \Phi \Rightarrow \Psi \mid \forall x \leq t.\Phi \mid \exists x \leq t.\Phi \mid \forall X.\Phi.$$

We then define the sets of formulas Σ_1^0 and Π_2^0 as follows:

- $A \in \Sigma_1^0$ iff $A \dashv\vdash_{\mathrm{K}} \exists x.B$ with $B \in \Delta_0^0$;
- $A \in \Pi_2^0$ iff $A \dashv\vdash_{\mathrm{K}} \forall x.B$ with $B \in \Sigma_1^0$.

Lemma 27 (Gödel) *If $A \in \Delta_0^0$ and $\vec{x} = x_1, \ldots, x_n$ are the free first-order variables of A, then there exists a function symbol f_A of arity n such that*

$$\mathbf{PA_2} \vdash A \Leftrightarrow f_A(\vec{x}) = 0.$$

Proof. The idea is that the truth value of a Δ_0^0 formula is primitive recursive. Formally, we take the integer 0 to mean "true" and any non-null integer to mean "false" and we consider the following primitive recursive definitions:

$$\begin{aligned} \mathrm{eq}(x,y) &= (x-y)+(y-x) \\ \mathrm{imp}(0,y) &= y \\ \mathrm{imp}(x+1,y) &= 0 \\ \mathrm{all}_f(0,\vec{y}) &= 0 \\ \mathrm{all}_f(x+1,\vec{y}) &= f(x,\vec{y}) + \mathrm{all}_f(x,\vec{y}) \\ \mathrm{one}_f(0,\vec{y}) &= 1 \\ \mathrm{one}_f(x+1,\vec{y}) &= f(x,\vec{y}) \cdot \mathrm{one}_f(x,\vec{y}) \end{aligned}$$

for any primitive recursive f. The intended meaning of the above functions is obvious: eq tests whether two integers are equal (in its definition we used the "truncated difference" and the addition functions, which are obviously primitive

recursive); the function imp implements the truth table of implication; $\mathrm{all}_f(x,\vec{y})$ checks whether $f(n,\vec{y})=0$ for all $0\leq n<x$, whereas $\mathrm{one}_f(x,\vec{y})$ checks whether there exists $0\leq n<x$ such that $f(n,\vec{y})=0$ (in the definitions of these functions we used addition and multiplication). By assumption, the function symbols eq, imp, all_f and one_f are available in the language of $\mathbf{PA_2}$ (these last two symbols are parametric in a term f which represents a primitive recursive function).

In what follows, we fix one first-order variable c_X for each second-order variable X. Let Φ be a formula belonging to the fragment defined above to introduce Δ^0_0 formulas, with free first-order variables $\vec{x}=x_1,\ldots,x_n$ and free second-order variables $X_1,\ldots,X_m$ (note that Φ is not necessarily second-order closed). We define a term $v_\Phi(\vec{x})$, containing the free variables $\vec{x}$, by induction on Φ:

$$\begin{array}{rcl} v_{t=u}(\vec{x}) & := & \mathrm{eq}(t,u)\\ v_X(\vec{x}) & := & c_X\\ v_{\Phi'\Rightarrow\Psi}(\vec{x}) & := & \mathrm{imp}(v_{\Phi'}(\vec{x}),v_\Psi(\vec{x}))\\ v_{\forall y\leq t.\Psi}(\vec{x}) & := & \mathrm{all}_{v_\Psi}(\mathsf{S}t,\vec{x})\\ v_{\exists y\leq t.\Psi}(\vec{x}) & := & \mathrm{one}_{v_\Psi}(\mathsf{S}t,\vec{x})\\ v_{\forall X.\Psi}(\vec{x}) & := & v_\Psi(\vec{x})[\mathsf{0}/c_X]+v_\Psi(\vec{x})[\mathsf{S0}/c_X] \end{array}$$

It is now enough to prove that

$$\mathbf{PA_2}, X_1\Leftrightarrow(c_{X_1}=0),\ldots,X_m\Leftrightarrow(c_{X_m}=0)\vdash\Phi\Leftrightarrow(v_\Phi=0).$$

In fact, by definition $v_\Phi(\vec{x})$ represents a primitive recursive function, hence there is already a function symbol f defining it, and the result follows from the fact that, by hypothesis, $\mathbf{PA_2}\vdash A\Leftrightarrow\Phi$ with Φ second-order closed (hence $m=0$).

The proof, which is by induction on Φ, is intuitively immediate (given the definition of v_Φ), but is long and tedious. It uses in an essential way induction (axiom I) and classical logic, *i.e.*, the elimination of double negation rule. We prefer to omit the details here. □

Theorem 28 (Friedman) *If A is a Σ^0_1 or Π^0_2 formula, then $\mathbf{PA_2}\vdash A$ iff $\mathbf{HA_2}\vdash A$.*

Proof. Let $A\in\Pi^0_2$ be such that $\mathbf{PA_2}\vdash A$. By definition and Lemma 27, we may suppose A to be of the form $\forall\vec{x}.\exists\vec{y}.f(\vec{x},\vec{y})=\mathsf{0}$. By considering a primitive recursive encoding of tuples, we may actually assume A to be of the even simpler form $\forall x.\exists y.f(x,y)=\mathsf{0}$. Let now R denote the formula $\exists y.f(x,y)=\mathsf{0}$. By applying a first-order universal quantification elimination rule to the proof of A, we obtain $\mathbf{PA_2}\vdash R$. By Theorem 21 and Lemma 25, we have $\mathbf{HA_2}\vdash R^R$. Now, $R^R=\neg_R\neg_R\exists y.\neg_R\neg_R(f(x,y)=\mathsf{0})$, so by points 2 and 5 of Lemma 17 we have $\mathbf{HA_2}\vdash\neg_R\forall y.\neg_R(f(x,y)=\mathsf{0})$. But this latter formula intuitionistically implies R, as the following derivation shows:

$$\dfrac{\neg_R\forall y.\neg_R(f(x,y)=\mathsf{0}) \qquad \dfrac{\dfrac{\dfrac{[f(x,y)=\mathsf{0}]^*}{R}}{\neg_R(f(x,y)=\mathsf{0})}\,{}^*}{\forall y.\neg_R(f(x,y)=\mathsf{0})}}{R}$$

Note that the first rule on the top-right of the derivation is a first-order existential quantifier introduction, with witness y, which allows us to obtain $\exists y.f(x,y) = \mathsf{0}$ (that is, the formula R itself) from the assumption $f(x,y) = \mathsf{0}$.

So we just proved that $\mathbf{HA_2} \vdash R$, which shows equiprovability of Σ^0_1 formulas. To conclude, we simply use a first-order universal quantifier introduction rule on R, which allows us to prove $\mathbf{HA_2} \vdash A$, as desired. □

Therefore, if we restrict to theorems of the form $\exists x.f(x) = \mathsf{0}$ or $\forall x.\exists y.f(x,y) = \mathsf{0}$, we do not even need to resort to adding double negations: Peano arithmetic simply does not prove anything more than Heyting arithmetic. In particular, $\mathbf{HA_2}$ and $\mathbf{PA_2}$ prove the totality of exactly the same set of recursive functions.

5.3 The hardness of proving normalization of system F

In this section we shall justify why the normalization of system **F** is a "strong" result. In fact, we shall see that it implies the consistency of $\mathbf{PA_2}$, which is an extremely powerful system (as we said above, most current mathematics, including analysis, may be formalized in $\mathbf{PA_2}$ plus some form of choice).

The proof, which we shall not give in full detail, is based on the following facts. First of all, standard proof-theoretic encodings (based on Gödelization) assure us that the following statements may be expressed as formulas of $\mathbf{PA_2}$:

$$\begin{array}{rcl} \mathrm{CE}(S) & := & \text{"the sequent calculus } S \text{ enjoys cut-elimination"}; \\ \mathrm{Con}(T) & := & \text{"the theory } T \text{ is consistent"}; \end{array}$$

where S is a generic sequent calculus system (with finitely many finite rules) and T is a generic theory on the language of $\mathbf{PA_2}$. Then, we have:

1. the complexity of cut-elimination of $\mathbf{LJ}_2$ (intuitionistic predicate second-order sequent calculus) is essentially contained in its *propositional* second-order fragment $\mathbf{LJ_F}$ ($\mathbf{LJ}_2$ without first order). Note that $\mathbf{LJ}_2$ is the sequent calculus in which $\mathbf{HA_2}$ may be formulated, whereas $\mathbf{LJ_F}$ is the sequent calculus corresponding to system **F**. Technically, one can prove that cut-elimination in $\mathbf{LJ_F}$ implies cut-elimination in $\mathbf{LJ}_2$. Furthermore, the proof is formalizable in $\mathbf{PA_2}$:

$$\mathbf{PA_2} \vdash \mathrm{CE}(\mathbf{LJ_F}) \Rightarrow \mathrm{CE}(\mathbf{LJ}_2).$$

2. Cut-elimination of $\mathbf{LJ}_2$ is such a strong statement that, by assuming it, one reduces the consistency of $\mathbf{HA_2}$ to the consistency of a much weaker system, namely *intuitionistic elementary arithmetic* (**IEA**). Of course, this may again be formalized in $\mathbf{PA_2}$:

$$\mathbf{PA_2} \vdash \mathrm{CE}(\mathbf{LJ}_2) \Rightarrow \mathrm{Con}(\mathbf{IEA}) \Rightarrow \mathrm{Con}(\mathbf{HA_2}).$$

3. The system **IEA** is actually weak enough that $\mathbf{PA_2}$ is able to prove it is consistent:

$$\mathbf{PA_2} \vdash \mathrm{Con}(\mathbf{IEA}).$$

Now, we know from Sect. 1.5 that normalization in natural deduction and cut-elimination in sequent calculus match each other, so normalization of system **F** (which is, via the Curry-Howard correspondence, a natural deduction system) is equivalent to cut-elimination of $\mathbf{LJ_F}$. We also know from Corollary 26 that the consistency of $\mathbf{HA_2}$ is actually equivalent to the consistency of $\mathbf{PA_2}$. Then, assuming that both of these last two results may also be formalized in $\mathbf{PA_2}$ (which is the case), expressing normalization of system **F** as a $\mathbf{PA_2}$ formula

$$\mathrm{N}(\mathbf{F}) := \text{“system } \mathbf{F} \text{ enjoys the normalization property”},$$

we have the following deduction entirely within $\mathbf{PA_2} + \mathrm{N}(\mathbf{F})$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overset{\mathbf{PA_2}}{\vdots\ 2}}{\mathrm{CE}(\mathbf{LJ}_2) \Rightarrow \mathrm{Con}(\mathbf{IEA}) \Rightarrow \mathrm{Con}(\mathbf{HA_2})} \quad \dfrac{\overset{\mathbf{PA_2}}{\vdots\ 1}}{\mathrm{CE}(\mathbf{LJ_F}) \Rightarrow \mathrm{CE}(\mathbf{LJ}_2)} \quad \dfrac{\dfrac{\overset{\mathbf{PA_2}}{\vdots\ \text{Sect. 1.5}}}{\mathrm{N}(\mathbf{F}) \Rightarrow \mathrm{CE}(\mathbf{LJ_F})} \quad \mathrm{N}(\mathbf{F})}{\mathrm{CE}(\mathbf{LJ_F})}}{\mathrm{Con}(\mathbf{IEA}) \Rightarrow \mathrm{Con}(\mathbf{HA_2}) \qquad \mathrm{CE}(\mathbf{LJ}_2)} \quad \dfrac{\overset{\mathbf{PA_2}}{\vdots\ 3}}{\mathrm{Con}(\mathbf{IEA})}}{\mathrm{Con}(\mathbf{HA_2})} \quad \dfrac{\overset{\mathbf{PA_2}}{\vdots\ \text{Corollary 26}}}{\mathrm{Con}(\mathbf{HA_2}) \Rightarrow \mathrm{Con}(\mathbf{PA_2})}}{\mathrm{Con}(\mathbf{PA_2})}$$

So adding the axiom $\mathrm{N}(\mathbf{F})$ (normalization of system **F**) to $\mathbf{PA_2}$ gives a theory proving the consistency of $\mathbf{PA_2}$. Therefore, by Gödel's second incompleteness theorem, the normalization of system **F** cannot itself be proved within $\mathbf{PA_2}$.[5]

Let us give some details about points 1, 2 and 3 above. For point 1, we define an erasure operation from formulas of $\mathbf{PA_2}$ to system **F** formulas, which “forgets” first order information, as follows:

$$\begin{aligned}
(t = u)^- &:= U \\
((t_1, \ldots, t_n) \in X)^- &:= X \\
(A \Rightarrow B)^- &:= A^- \Rightarrow B^- \\
(\forall x.A)^- &:= A^- \\
(\exists x.A)^- &:= A^- \\
(\forall X.A)^- &:= \forall X.A^-
\end{aligned}$$

where U is any closed system **F** formula.

We extend the erasure operation to sequent calculus derivations: we associate a derivation δ^- of $\Gamma^- \vdash \Sigma^-$ in $\mathbf{LJ_F}$ with every derivation δ of $\Gamma \vdash \Sigma$ in $\mathbf{LJ}_2$, which is obtained from δ by erasing all first-order rules.

Lemma 29 *Let δ be a derivation in $\mathbf{LJ}_2$.*

1. *if δ^- is cut-free, then δ is cut-free;*
2. *if $\delta^- \to \gamma$ by means of a cut-elimination step, then $\delta \to^+ \delta_1$ such that $\delta_1^- = \gamma$.*

[5]Note that it is essential that the implication $\mathrm{N}(\mathbf{F}) \Rightarrow \mathrm{Con}(\mathbf{PA_2})$ be provable in $\mathbf{PA_2}$ (or a weaker system). If it were provable is a stronger system, we would not be able to conclude anything about the ability of $\mathbf{PA_2}$ to prove normalization of system **F**. We hope the reader to be convinced of the possibility of formalizing all the needed results (such as Proposition 30 and Proposition 35 below), even though such a formalization will not be discussed here.

PROOF. Point 1 is obvious. Point 2 is analogous to points 3 and 4 of Lemma 38 below, but more tedious, because of all the commutations of sequent calculus. We omit the details. □

Proposition 30 *Cut-elimination in* $\mathbf{LJ_F}$ *implies cut-elimination in* $\mathbf{LJ}_2$.

PROOF. Let δ be a proof in $\mathbf{LJ}_2$. Since $\mathbf{LJ}_F$ enjoys cut-elimination, we have $\delta^- \to^* \gamma_0$ with γ_0 cut-free. By iterating point 2 of Lemma 29, we obtain $\delta \to^* \delta_0$ such that $\delta_0^- = \gamma_0$. But by point 1 of Lemma 29, δ_0 is cut-free. □

For what concerns points 2 and 3, let us first define the system **IEA**. Its language of terms is the same as $\mathbf{PA_2}$, but its formulas are restricted to first order (and therefore, for reasons of expressiveness, conjunction, disjunction and absurdity must be primitive connectives). The axioms of **IEA** are

$$\begin{aligned}
E_r &:= \forall x.x = x,\\
E_c[t,u] &:= \forall xy.x = y \Rightarrow (t[x/z] = u[x/z]) \Rightarrow (t[y/z] = u[y/z])\\
&\qquad \text{for all terms } t \neq u \text{ s.t. } z \in \mathrm{fv}(t) \cup \mathrm{fv}(u),\\
E_{pr} &:= \text{universally closed equations defining primitive recursive functions},\\
P_4 &:= \forall x.\mathsf{S}x \neq \mathsf{0}.
\end{aligned}$$

The axioms of the form $E_c[t,u]$ basically say that equality implies Leibniz equality instantiated on atomic formulas. From these axioms and E_r all the basic properties of equality (symmetry, transitivity, contextuality) may be derived. We take the proof system of **IEA** to be first-order **LJ** (but of course first-order **NJ** would be equivalent). To sum up, considering that the axioms of the form $E_c[t,u]$ are all instances of E_L, the relationship between **IEA** and $\mathbf{HA_2}$ is

$$\mathbf{HA_2} \quad = \quad \mathbf{IEA} + E_L + I + \text{comprehension scheme}.$$

The comprehension scheme is implicitly added in shifting from **LJ** to $\mathbf{LJ}_2$.

We shall not give any detail about point 3, except saying that the consistency of **IEA**, which is non-trivial only because of P_4, may actually be proved in a much weaker system, known as *primitive recursive arithmetic* (**PRA**). We refer the reader to [Gir87] for a proof.

To prove point 2, we introduce two transformations on $\mathbf{PA_2}$ formulas, denoted by $(\cdot)^\bullet$ and $(\cdot)^\circ$. For the second one, we use the formula

$$\mathsf{CL}(X) := \forall xy.x = y \Rightarrow x \in X \Rightarrow y \in X,$$

which states that "the set X is extensional". The translations are defined by induction. Both are the identity on atomic formulas; for composite formulas, we set:

$$\begin{array}{rclcrcl}
(A \Rightarrow B)^\bullet &:=& A^\bullet \Rightarrow B^\bullet & \qquad & (A \Rightarrow B)^\circ &:=& A^\circ \Rightarrow B^\circ,\\
(\forall x.A)^\bullet &:=& \forall x.\mathsf{Int}(x) \Rightarrow A^\bullet & & (\forall x.A)^\circ &:=& \forall x.A^\circ,\\
(\exists x.A)^\bullet &:=& \exists x.\mathsf{Int}(x) \wedge A^\bullet & & (\exists x.A)^\circ &:=& \exists x.A^\circ,\\
(\forall X.A)^\bullet &:=& \forall X.A^\bullet & & (\forall X.A)^\circ &:=& \forall X.\mathsf{CL}(X) \Rightarrow A^\circ.
\end{array}$$

So the translation $(\cdot)^\bullet$ relativizes first-order quantifiers to integers and the translation $(\cdot)^\circ$ relativizes second-order quantifiers to extensional sets. The main properties of the translations are given below.

Lemma 31 *1. If $\Gamma \vdash \Sigma$ is derivable in $\mathbf{LJ}_2$ and if $x_1, \ldots, x_m$ are the free first-order variables appearing in Γ, Σ, then*

$$\mathbf{IEA}, E_L, \mathsf{Int}(x_1), \ldots, \mathsf{Int}(x_m), \Gamma^\bullet \vdash \Sigma^\bullet$$

is derivable in $\mathbf{LJ}_2$, where $\mathbf{IEA}$ on the left means, as usual, a finite number of axioms of $\mathbf{IEA}$;

2. if A is an axiom of $\mathbf{IEA}$ or E_L, we have $A \vdash A^\bullet$ in $\mathbf{LJ}_2$;

3. $\mathbf{IEA}, E_L \vdash I^\bullet$ is derivable in $\mathbf{LJ}_2$.

(We remind that E_L and I are the two axioms of $\mathbf{PA_2}$ and $\mathbf{HA_2}$ which are not axioms of $\mathbf{IEA}$).

Proof. We start with observing that point 2 is immediate: all axioms are universally quantified without any relativization, and a non-relativized statement trivially implies its relativized version.

For point 1 and 3, we are going to need the following result:

Lemma 32 *If t is any term whose free variables are among $x_1, \ldots, x_m$, then*

$$\mathbf{IEA}, E_L, \mathsf{Int}(x_1), \ldots, \mathsf{Int}(x_m) \vdash \mathsf{Int}(t)$$

is derivable in $\mathbf{LJ}_2$. □

The reason why Lemma 32 holds is that t only contains symbols for primitive recursive functions, which are equationally defined by the axioms of **IEA**. The axiom E_L is needed to use the equations and the formulas of the form $\mathsf{Int}(x_i)$ allow us to apply induction. We invite the reader to do the proof, which is a quite instructive programming exercise.[6]

The proof of point 1 is by induction on the last rule of the proof. The only interesting cases are the left universal quantifier rule and the right existential quantifier rule. We shall treat the first, leaving the other to the reader. So suppose that our derivation ends with

$$\dfrac{\begin{array}{c}\vdots\ \delta \\ \Gamma, A[t/z] \vdash \Sigma\end{array}}{\Gamma, \forall z.A \vdash \Sigma}\,.$$

We may further assume that every free variable of t is also free in Γ or Σ. In fact, if there is $y \in \mathrm{fv}(t)$ which does not appear in Γ, Σ, then such a variable may be replaced by anything (it is "generic"), in particular by a closed term like $\mathsf{0}$. This means that we may consider the proof

$$\dfrac{\begin{array}{c}\vdots\ \delta[\mathsf{0}/y] \\ \Gamma, A[t[\mathsf{0}/y]/z] \vdash \Sigma\end{array}}{\Gamma, \forall z.A \vdash \Sigma}$$

[6] Basically, proving Lemma 32 amounts to programming primitive recursive functions in system **F**, as we did in Sect. 4.3. In fact, erasing all first-order information from a cut-free derivation of the sequent $\mathbf{IEA}, E_L, \mathsf{Int}(x_1), \ldots, \mathsf{Int}(x_m) \vdash \mathsf{Int}(t)$ and converting it to a natural deduction proof yields a system **F** term which computes exactly the m-ary primitive recursive function described by $t(\vec{x})$. See [Kri93] for a proof of this fact.

instead, which satisfies our assumption and still allows us to apply induction to $\delta[\mathsf{0}/z]$, which is structurally identical to δ. So the induction hypothesis applied to δ gives us a proof of

$$\mathbf{IEA}, E_L, \mathsf{Int}(\vec{x}), \Gamma^\bullet, A^\bullet[t/z] \vdash \Sigma^\bullet,$$

where $\vec{x}$ are the free variables of Γ, Σ (and, by our assumption, they also contain those of t) and $\mathsf{Int}(\vec{x})$ is a shorthand for all the formulas of the form $\mathsf{Int}(x_i)$. (We also implicitly used the immediate fact that $(A[t/z])^\bullet = A^\bullet[t/z]$). From this, using Lemma 32, we have

$$\dfrac{\dfrac{\mathbf{IEA}, E_L, \mathsf{Int}(\vec{x}) \vdash \mathsf{Int}(t/z) \quad \mathbf{IEA}, E_L, \mathsf{Int}(\vec{x}), \Gamma^\bullet, A^\bullet[t/z] \vdash \Sigma^\bullet}{\mathbf{IEA}, E_L, \mathsf{Int}(\vec{x}), \Gamma^\bullet, \mathsf{Int}(t/z) \Rightarrow A^\bullet[t/z] \vdash \Sigma^\bullet}}{\mathbf{IEA}, E_L, \mathsf{Int}(\vec{x}), \Gamma^\bullet, (\forall z.A)^\bullet \vdash \Sigma^\bullet}$$

where we implicitly used some contraction rules.

For what concerns point 3, call γ the following derivation:

$$\dfrac{\dfrac{\vdots \text{ Lemma 32}}{\mathbf{IEA}, E_L, \mathsf{Int}(y) \vdash \mathsf{Int}(\mathsf{S}y)} \quad \dfrac{\dfrac{\dfrac{}{\mathsf{Int}(y) \vdash \mathsf{Int}(y)} \quad \dfrac{\overline{y \in X \vdash y \in X} \quad \overline{\mathsf{S}y \in X \vdash \mathsf{S}y \in X}}{y \in X \Rightarrow \mathsf{S}y \in X, y \in X \vdash \mathsf{S}y \in X}}{\mathsf{Int}(y) \Rightarrow y \in X \Rightarrow \mathsf{S}y \in X, \mathsf{Int}(y), y \in X \vdash \mathsf{S}y \in X}}{(\forall y.y \in X \Rightarrow \mathsf{S}y \in X)^\bullet, \mathsf{Int}(y), y \in X \vdash \mathsf{S}y \in X}}{\dfrac{\mathbf{IEA}, E_L, (\forall y.y \in X \Rightarrow \mathsf{S}y \in X)^\bullet, \mathsf{Int}(y), \mathsf{Int}(y), y \in X \vdash \mathsf{Int}(\mathsf{S}y) \wedge \mathsf{S}y \in X}{\mathbf{IEA}, E_L, (\forall y.y \in X \Rightarrow \mathsf{S}y \in X)^\bullet \vdash \forall y.(\mathsf{Int}(y) \wedge y \in X) \Rightarrow (\mathsf{Int}(\mathsf{S}y) \wedge \mathsf{S}y \in X).}}$$

Now, if we let

$$\begin{aligned} A &:= (\forall y.y \in X \Rightarrow \mathsf{S}y \in X)^\bullet, \\ B &:= \forall y.(\mathsf{Int}(y) \wedge y \in X) \Rightarrow (\mathsf{Int}(\mathsf{S}y) \wedge \mathsf{S}y \in X), \\ \mathsf{Int}_0(X, x) &:= \mathsf{0} \in X \Rightarrow (\forall y.y \in X \Rightarrow \mathsf{S}y \in X) \Rightarrow x \in X, \end{aligned}$$

(so that γ is of conclusion $\mathbf{IEA}, E_L, A \vdash B$), we invite the reader to check that

$$\begin{aligned} \mathsf{Int}(x) &= \forall X.\mathsf{Int}_0(X, x), \\ \mathsf{Int}(x)^\bullet &= \forall X.\mathsf{0} \in X \Rightarrow A \Rightarrow x \in X, \\ \mathsf{Int}_0(X, x)[\mathsf{Int}(z) \wedge z \in X/Xz] &= (\mathsf{Int}(\mathsf{0}) \wedge \mathsf{0} \in X) \Rightarrow B \Rightarrow (\mathsf{Int}(x) \wedge x \in X). \end{aligned}$$

With these notations in place, the proof of point 3 consists of the following derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots \text{ Lemma 32}}{\mathbf{IEA}, E_L \vdash \mathsf{Int}(\mathsf{0})} \quad \dfrac{}{\mathsf{0} \in X \vdash \mathsf{0} \in X}}{\mathbf{IEA}, E_L, \mathsf{0} \in X \vdash \mathsf{Int}(\mathsf{0}) \wedge \mathsf{0} \in X} \quad \dfrac{\vdots\ \gamma}{\mathbf{IEA}, E_L, A \vdash B} \quad \dfrac{\overline{x \in X \vdash x \in X}}{\mathsf{Int}(x) \wedge x \in X \vdash x \in X}}{\dfrac{\mathbf{IEA}, E_L, \mathsf{0} \in X \vdash \mathsf{Int}(\mathsf{0}) \wedge \mathsf{0} \in X \quad \mathbf{IEA}, E_L, B \Rightarrow (\mathsf{Int}(x) \wedge x \in X), A \vdash x \in X}{\mathbf{IEA}, \mathbf{IEA}, E_L, E_L, (\mathsf{Int}(\mathsf{0}) \wedge \mathsf{0} \in X) \Rightarrow B \Rightarrow (\mathsf{Int}(x) \wedge x \in X), \mathsf{0} \in X, A \vdash x \in X}}{\dfrac{\mathbf{IEA}, \mathbf{IEA}, E_L, E_L, \mathsf{Int}(x), \mathsf{0} \in X, A \vdash x \in X}{\mathbf{IEA}, E_L, \mathsf{Int}(x) \vdash \mathsf{Int}(x)^\bullet}}{\dfrac{\mathbf{IEA}, E_L, \mathsf{Int}(x) \vdash \mathsf{Int}(x)^\bullet}{\mathbf{IEA}, E_L \vdash \mathsf{Int}(x) \Rightarrow \mathsf{Int}(x)^\bullet}}{\mathbf{IEA}, E_L \vdash I^\bullet.}}$$

□

Lemma 33 *1. If $\Gamma \vdash \Sigma$ is derivable in $\mathbf{LJ}_2$ and if $X_1, \ldots, X_n$ are the free second-order variables appearing in Γ, Σ, then*

$$\mathbf{IEA}, \mathsf{CL}(X_1), \ldots, \mathsf{CL}(X_n), \Gamma^\circ \vdash \Sigma^\circ$$

*is derivable in $\mathbf{LJ}_2$, where **IEA** on the left means, as usual, a finite number of axioms of **IEA**;*

2. $\vdash E_L^\circ$ is derivable in $\mathbf{LJ}_2$;

PROOF. Point 1 is similar to point 1 of Lemma 31: it is proved by induction on the last rule of the proof and the only interesting case is the left second-order quantifier rule, which needs the following result, whose proof we leave to the reader:

Lemma 34 *For every formula B containing the free first-order variable z and whose free second-order variables are contained in $X_1, \ldots, X_n$, the sequent $\mathbf{IEA}, \mathsf{CL}(X_1), \ldots, \mathsf{CL}(X_n) \vdash \mathsf{CL}(Z)[B^\circ/Zz]$ is derivable in $\mathbf{LJ}_2$.* □

For what concerns point 2, the proof is given by the following derivation:

$$\dfrac{\dfrac{\dfrac{}{x = y \vdash x = y} \quad \dfrac{\dfrac{}{x \in Z \vdash x \in Z} \quad \dfrac{}{y \in Z \vdash y \in Z}}{x \in Z \Rightarrow y \in Z, x \in Z \vdash y \in Z}}{x = y, x = y \Rightarrow x \in Z \Rightarrow y \in Z, x \in Z \vdash y \in Z}}{x = y, \mathsf{CL}(Z), x \in Z \vdash y \in Z}}{\vdash E_L^\circ}$$

□

Proposition 35 *Assuming cut-elimination of $\mathbf{LJ}_2$, the consistency of **IEA** implies the consistency of $\mathbf{HA_2}$.*

PROOF. We prove the contrapositive implication. Suppose that $\mathbf{HA_2}$ is inconsistent. This means that we have a proof in $\mathbf{LJ}_2$ of a sequent of the form

$$\mathbf{IEA}, E_L, I \vdash .$$

By point 1 of Lemma 31, we have a proof, still in $\mathbf{LJ}_2$, of

$$\mathbf{IEA}, E_L, \mathbf{IEA}^\bullet, E_L^\bullet, I^\bullet \vdash .$$

Note that there are no formulas of the form $\mathsf{Int}(x)$ on the left because all axioms of **IEA** and $\mathbf{HA_2}$ are first-order closed. From this, using point 2 of Lemma 31, we may remove the $(\cdot)^\bullet$ from the axioms of **IEA** and from E_L, and using point 3 we may remove altogether the formula $I^\bullet$, obtaining a proof of the sequent

$$\mathbf{IEA}, E_L \vdash .$$

We now apply point 1 of Lemma 33, which gives us a proof, still in $\mathbf{LJ}_2$, of the sequent

$$\mathbf{IEA}, \mathbf{IEA}^\circ, E_L^\circ \vdash .$$

Again, note that there are no formulas of the form $\mathsf{CL}(X)$ on the left because all axioms of **IEA** and E_L are second-order closed. Point 2 of Lemma 33 allows

us to remove the $(\cdot)^\circ$ from the axioms of **IEA** and point 3 allows us to remove the formula E_L° altogether, obtaining a proof of

$$\mathbf{IEA} \vdash .$$

Although this shows that the axioms of **IEA** yield a contradiction in $\mathbf{LJ}_2$, we cannot conclude yet: the proof contains cuts which use second-order formulas, so the contradiction might come from the second-order rules of $\mathbf{LJ}_2$ and not from the axioms of **IEA**.

This is where we use our hypothesis that $\mathbf{LJ}_2$ enjoys cut-elimination: after eliminating the cuts, we know that the proof verifies the subformula property, and since no axiom of **IEA** uses second order, we actually know that the proof is in **LJ** and, therefore, **IEA** really is inconsistent. □

6 Realizability and strong normalization of system F

The reducibility argument we gave in Sect. 4.6 is actually a weakened version of the original argument of Girard [GLT89], who proved the *strong* normalization of system **F**, *i.e.*, that no term of system **F** admits an infinitely long reduction sequence. In other words, no matter what strategy we use to normalize terms, we will always reach the normal form.

Here, we shall prove strong normalization of system **F** using a related, but slightly different technique, called *realizability*. Realizability was introduced by Kleene in the study of models of intuitionistic logic. The formulation we use here is due to Krivine, who extended it to classical logic [Kri04] (and for much broader uses than just proving strong normalization).

6.1 "Curry-style" system F

The definition of system **F** given in Sect. 3.2 (and also that of the simply typed λ-calculus given in Sect. 2) is known as "Church-style". Its peculiarity is that the typing information is incorporated in the term, so that each term has a unique type. Alternatively, the construction of system **F** terms (or simply typed terms) may be seen as a "decoration" of natural deduction with pure λ-terms, yielding what is known as "Curry-style" system **F** or, more generally, a *typing system*. A typing system attempts to assign a type to *pure* λ-terms, *i.e.*, λ-terms containing no type annotation. In general, not all terms will be typable, and the typable terms will be guaranteed to enjoy special properties (for instance, some form of program correctness).

In Curry-style system **F**, we no longer use a fixed type-assignment function from variables to types, as we did in Church-style system **F**. Instead, a variable is assigned a type "locally", by means of a *variable type assignment* of the form $x : A$. A *context* is a finite set Γ of type assignments, such that every variable appears in Γ with at most one type.

The typing system is a calculus for deriving *judgments*, which are objects of the form $\Gamma \vdash M : A$, to be read "the (pure) term M has type A under the

context Γ". The rules for forming typing judgments are exactly those of natural deduction:

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \Rightarrow B} \qquad \frac{\Gamma \vdash M : A \Rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall X.A}\ {}^{\dagger} \qquad \frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash M : A[B/X]}$$

Condition (†) is the usual one: X is not free in any type appearing in Γ. A term M such that $\Gamma \vdash M : A$ may be derived in the above system for some Γ and some A is said to be **F**-*typable*.

Note how the typing rules follow exactly the definition of Church-style system **F** terms. However, there is no trace of the typing in pure λ-terms; even worse, universal quantification is "invisible" in terms. Hence, it is impossible to speak of "the" type of a term in Curry-style system **F**; in fact, an **F**-typable term always has infinitely many types (for instance, $\lambda x.x$ has every type of the form $A \Rightarrow A$, for all A, together of course with the type $\forall X.X \Rightarrow X$, which subsumes them all).

The Church-style formulation gives an exact Curry-Howard correspondence; the Curry-style formulation is important for practical purposes, *i.e.*, designing functional programming languages. For instance, "real world" languages such as ML and Haskell are based on weakened versions of Curry-style system **F**.[7]

6.2 Comparing the two formulations

The two formulations of system **F** are nevertheless closely related. An important tool in studying their relationship is the *erasure* $(\cdot)^-$ of Church-style terms into pure λ-terms, defined inductively as follows:

$$\begin{aligned}
(x^A)^- &= x \\
(\lambda x^A.M)^- &= \lambda x.M^- \\
(MN)^- &= M^- N^- \\
(\Lambda X.M)^- &= M^- \\
(MA)^- &= M^-
\end{aligned}$$

Erasure is of course compatible with substitution:

Lemma 36 *For all system* **F** *terms* M, N *and variable* x *of suitable types,* $(M[N/x])^- = M^-[N^-/x]$.

Proof. A straightforward induction on M. □

The first relation between the two formulations is given by the following:

[7] In fact, it may be proved that it is undecidable whether a term is typable in the above system [Wel99]. By contrast, the typing system of a concrete functional programming language must be decidable, in order to have an effective type-inference mechanism.

Proposition 37 *1. Let $M : A$ be a system* **F** *term with* $\mathrm{fv}(M) = \{x_1, \ldots, x_n\}$*, of respective types $B_1, \ldots, B_n$. Then, the judgment $x_1 : B_1, \ldots, x_n : B_n \vdash M^- : A$ is derivable.*

2. Conversely, if δ is a derivation of the judgment $\Gamma \vdash M : A$, it is obviously isomorphic to a natural deduction proof of A from hypotheses included in the types of Γ, which itself is isomorphic to a system **F** *term M_δ of type A and free variables among the ones appearing in Γ; then, we actually have $M_\delta^- = M$.*

Proof. Both points are proved by induction on M. □

So each system **F** term induces an **F**-typable λ-term, and each type derivation induces a system **F** term whose erasure is precisely the typed λ-term.

The two formulations are also equivalent in terms of normalization properties. In the sequel, we shall call *universal reduction step* a reduction in a system **F** term involving a redex of the form $(\Lambda X.M)A$.

Lemma 38 *Let M be a system* **F** *term. Then:*

1. *M normal implies M^- normal;*
2. *$M \to N$ implies $N^- = M^-$ if the reduction step is universal, or $M^- \to N^-$ otherwise;*
3. *M^- normal implies that all reductions starting from M contain only universal steps.*
4. *$M^- \to K$ implies $M \to^+ N$ such that $N^- = K$.*

Proof. Points (1) and (2) are obvious: universal redexes and reduction steps correspond to nothing at all in the erasure, while the other redexes and reduction steps are exactly β-redexes and β-reductions, respectively.

Point (3), is proved by contraposition. Suppose that $M \to^* M_1 \to M_2$ with the last reduction involving a non-universal redex. By point (2), we have $M^- = M_1^- \to M_2^-$, so M^- is not normal.

For point (4), we proceed by induction on M. The only interesting case is application: $M = M_1 M_2$. There are two cases:

i. the redex reduced in M^- is in M_1^- or M_2^-;

ii. M^- is itself a redex, *i.e.*, $M_1^- = \lambda x.M_0^-$.

The first case is straightforwardly handled by induction. In the second case, M_1 might be more complex than an abstraction; in fact, its general form is

$$M_1 = (\Lambda X_1 \ldots X_n . \lambda x^A . M_0) B_1 \cdots B_n.$$

Other forms are excluded by the typing constraints and by the definition of erasure. Then, the one-step reduction in the pure λ-calculus is simulated by several steps in system **F**, using Lemma 36. □

Lemma 39 *For all system* **F** *term M, there is no infinite reduction sequence $M \to M_1 \to M_2 \to \cdots$ consisting entirely of universal reduction steps.*

PROOF. Define inductively the *type-free size* $\#(\cdot)$ of system **F** terms, as follows:

- $\#(x^A) = 1$;
- $\#(\lambda x^A.M) = \#(M) + 1$;
- $\#(MN) = \#(M) + \#(N) + 1$;
- $\#(\Lambda X.M) = \#(M)$;
- $\#(MA) = \#(M)$.

It is easy to see that, for every type A and system **F** term M, $\#(M[A/X]) = \#(M)$. From this, it follows that a universal reduction step shrinks the type-free size of terms by 2; hence, the result follows by induction on $\#(M)$. □

Proposition 40 *System* **F** *is (strongly) normalizing iff every* **F***-typable* λ*-term is (strongly) normalizable.*

PROOF. Let us us start with weak normalization. Suppose that system **F** is normalizing, and take an **F**-typable λ-term M. If δ is the derivation typing it, we may consider the induced system **F** term M_δ. We know that $M_\delta \to^* N$, with N normal. Then, we apply point 2 of Proposition 37 and point 2 of Lemma 38 to obtain $M \to^* N^-$, and N^- is normal by point 1 of Lemma 38.

Suppose now that every **F**-typable term is normalizable, and consider a system **F** term M. By hypothesis, $M^- \to^* K$, with K normal. By point 4 of Lemma 38, $M \to^* N$ with $N^- = K$. By point 3 of Lemma 38, every reduction of N only contains universal steps; but then N is normalizable by Lemma 39.

For strong normalization, we prove the contrapositive in both directions. Suppose there exists an **F**-typable λ-term M admitting an infinite reduction sequence. We know that the type derivation δ for M induces a system **F** term M_δ such that $M_\delta^- = M$. Then, by point (4) of Lemma 38, M_δ too has an infinite reduction sequence, so system **F** is not strongly normalizing.

Conversely, suppose there is a system **F** term M which is not strongly normalizing, and take an infinite reduction starting from it. By Lemma 39, such reduction contains infinitely many non-universal reduction steps; hence, by point (2) of Lemma 38, M^- is not strongly normalizable. □

6.3 The realizability technique

Thanks to Proposition 40, we may prove the strong normalization of system **F** by proving that every **F**-typable λ-term strongly normalizes. We do this using an argument due to Krivine [Kri93, Kri04], which generalizes the original proof of Girard [GLT89] into a powerful technique known as *realizability*.

Let Λ be the set of all pure λ-terms, and we denote by $\mathcal{V}$ the set of variables. We define Π as the set of finite sequences of λ-terms, which we call *stacks*. We use the notation $\vec{P}$ to range over stacks. Moreover, if $M \in \Lambda$ and $\vec{P} = \langle M_1, \ldots, M_n \rangle$, we define the notation $M \cdot \vec{P} = \langle M, M_1, \ldots, M_n \rangle$.

Definition 8 (Pole) *Let* Λ_0 *be an arbitrary set of* λ*-terms containing all variables, i.e., such that* $\mathcal{V} \subseteq \Lambda_0$*. A* Λ_0-pole *is a set* $\perp\!\!\!\perp \subseteq \Lambda \times \Pi$ *such that*

P1. $(M[N/x], \vec{P}) \in \perp\!\!\!\perp$ *and* $N \in \Lambda_0$ *implies* $(\lambda x.M, N \cdot \vec{P}) \in \perp\!\!\!\perp$*;*

P2. $(M, N \cdot \vec{P}) \in \perp\!\!\!\perp$ *implies* $(MN, \vec{P}) \in \perp\!\!\!\perp$*.*

Definition 9 (Falsity value, orthogonal) *A* falsity value *is any subset of* Π*. Given a* Λ_0*-pole* $\perp\!\!\!\perp$ *and* $F \subseteq \Pi$*, we define*

$$F^{\perp} = \{M \in \Lambda \mid \forall \vec{P} \in F,\ (M, \vec{P}) \in \perp\!\!\!\perp\}.$$

Definition 10 (Valuation, interpretation, realizability) *We let* $\mathcal{A}$ *be the set of propositional atoms (i.e., atomic types). A* valuation *is a function* $\rho : \mathcal{A} \to \wp(\Pi)$ *(where* $\mathcal{A}$ *is the set of propositional atoms). Given a valuation* ρ*,* $F \subseteq \Pi$*, and* $X \in \mathcal{A}$*, we define, a new valuation* $\rho[X := F]$ *as follows:*

$$\text{for all } Y \in \mathcal{A}, \quad \rho[X := F](Y) = \begin{cases} F & \text{if } Y = X, \\ \rho(Y) & \text{if } Y \neq X. \end{cases}$$

We fix a Λ_0*-pole* $\perp\!\!\!\perp$*, we denote by* Π_0 *the set of stacks whose* λ*-terms all belong to* Λ_0*, and we define* $\mathbb{F}_{\Lambda_0} = \{F \subseteq \Pi \mid \emptyset \subsetneq F \subseteq \Pi_0\}$*. Then, if* ρ *is a valuation, we define its induced* interpretation function $\| \cdot \|_\rho$ *by induction on the formulas of* $\mathbf{F}$*:*

- $\|X\|_\rho = \rho(X)$*;*
- $\|A \Rightarrow B\|_\rho = \{M \cdot \vec{P} \in \Pi \mid M \in \|A\|_\rho^{\perp},\ \vec{P} \in \|B\|_\rho\}$*;*
- $\|\forall X.A\|_\rho = \bigcup_{F \in \mathbb{F}_{\Lambda_0}} \|A\|_{\rho[X:=F]}$*.*

Given a Λ_0*-pole and a valuation* ρ*, we say that a* λ*-term* M realizes *a formula* A*, and write* $M \Vdash_\rho A$*, if* $M \in \|A\|_\rho^{\perp}$*.*

Observe that realizability has 2 parameters: the pole and the valuation. For simplicity, the notation mentions explicitly only this latter, but it must be clear that realizability also depends on the specific choice of pole.

The next result holds independently of all parameters, *i.e.*, the valuation mentioned in it is generic, as is the pole.

Lemma 41 (Substitution) *For every formulas* A, B *and propositional atom* X*,* $\|A[B/X]\|_\rho = \|A\|_{\rho[X:=\|B\|_\rho]}$*.*

Proof. By induction on A. For simplicity, we set $\rho' = \rho[X := \|B\|_\rho]$.

If A is a propositional atom, then either $A = X$, or $A = Y \neq X$. Both cases are obvious.

Let $A = A_1 \Rightarrow A_2$. Since $A[B/X] = A_1[B/X] \Rightarrow A_2[B/X]$, we have that $\vec{P} \in \|A[B/X]\|_\rho$ iff $\vec{P} = M_1 \cdot \vec{P}_2$ with $M_1 \in \|A_1[B/X]\|_\rho^{\perp}$ and $\vec{P}_2 \in \|A_2[B/X]\|_\rho$, which, applying twice the induction hypothesis, holds iff $M_1 \in \|A_1\|_{\rho'}^{\perp}$ and $\vec{P}_2 \in \|A_2\|_{\rho'}$, which holds iff $\vec{P} \in \|A\|_{\rho'}$.

Let $A = \forall Y.A_0$, where, by α-equivalence, we may always suppose $Y \neq X$. Then, since $A[B/X] = \forall Y.A_0[B/X]$, we have $\vec{P} \in \|A[B/X]\|_\rho$ iff $\vec{P} \in \|A_0[B/X]\|_{\rho[Y:=F]}$ for some $F \in \mathbb{F}_{\Lambda_0}$, which, applying the induction hypothesis and the fact that $\rho[Y := F][X := \|B\|_\rho] = \rho'[Y := F]$, holds iff $\vec{P} \in \|A_0\|_{\rho'[Y:=F]}$ for some $F \in \mathbb{F}_{\Lambda_0}$, which holds iff $\vec{P} \in \|A\|_{\rho'}$. □

Definition 11 (Adaptation, admissible set) *A valuation ρ is said to be* adapted *to a Λ_0-pole $\perp\!\!\!\perp$ if, for every formula A,*

$$\mathcal{V} \subseteq \|A\|_\rho^\perp \subseteq \Lambda_0$$

(where we remind that $\mathcal{V}$ is the set of all λ-calculus variables).

A set $\Lambda_0 \subseteq \Lambda$ is admissible *if there exists a Λ_0-pole and a valuation adapted to it.*

Lemma 42 (Adequacy) *Let ρ be a valuation adapted to a Λ_0-pole, let δ be a derivation of $x_1 : B_1, \ldots, x_n : B_n \vdash M : A$ in "Curry-style" system $\mathbf{F}$, and let $N_1, \ldots, N_n \in \Lambda$ be such that $N_i \Vdash_\rho B_i$, for all $1 \leq i \leq n$. Then, $M[\vec{N}/\vec{x}] \Vdash_\rho A$, where $M[\vec{N}/\vec{x}]$ denotes the term obtained by simultaneously substituting each N_i to x_i in M.*

Proof. We start by setting $\Gamma = \{x_1 : B_1, \ldots, x_n : B_n\}$, and $M' = M[\vec{N}/\vec{x}]$. The proof is by induction on the last rule of the derivation δ.

The case of an axiom is trivial.

Let δ end with

$$\frac{\Gamma, x : A_1 \vdash M_0 : A_0}{\Gamma \vdash \lambda x.M_0 : A_1 \Rightarrow A_0}$$

We set $M_0' = M_0[\vec{N}/\vec{x}]$. Let $\vec{P} \in \|A_1 \Rightarrow A_0\|_\rho$; by definition, $\vec{P} = N \cdot \vec{P_0}$, such that $N \Vdash_\rho A_1$ and $\vec{P_0} \in \|A_0\|_\rho$. But the induction hypothesis gives us that $M_0'[N/x] \Vdash_\rho A_0$, which means that $(M_0'[N/x], \vec{P_0}) \in \perp\!\!\!\perp$. Observe that $N \Vdash_\rho A_1$ implies, by adaptation, that $N \in \Lambda_0$, so by P1 we have $(M', \vec{P}) \in \perp\!\!\!\perp$; since $\vec{P}$ is generic, we have shown that $M' \Vdash_\rho A_1 \Rightarrow A_0$.

Let δ end with

$$\frac{\Gamma \vdash M_0 : A_1 \Rightarrow A \quad \Gamma \vdash M_1 : A_1}{\Gamma \vdash M_0 M_1 : A}$$

We set $M_0' = M_0[\vec{N}/\vec{x}]$ and $M_1' = M_1[\vec{N}/\vec{x}]$. Applying twice the induction hypothesis, we obtain $M_0' \Vdash_\rho A_1 \Rightarrow A$ and $M_1' \Vdash_\rho A_1$; then, if $\vec{P}$ is a generic element of $\|A\|_\rho$, we have $M_1' \cdot \vec{P} \in \|A_1 \Rightarrow A\|_\rho$, which implies $(M_0', M_1'\vec{P}) \in \perp\!\!\!\perp$, which, by P2, implies $(M', \vec{P}) \in \perp\!\!\!\perp$, which shows that $M' \Vdash_\rho A$.

Let δ end with

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall X.A}$$

Let $\vec{P} \in \|\forall X.A\|_\rho$. By definition, there exists $F \in \mathbb{F}$ such that $\vec{P} \in \|A\|_{\rho[X:=F]}$. But X does not appear free in any B_i, which means that $\|B_i\|_{\rho[X:=F]} = \|B_i\|_\rho$, so we have in fact $N_i \Vdash_{\rho[X:=F]} B_i$, for all $1 \leq i \leq n$. Then, by the induction hypothesis, we have $M' \Vdash_{\rho[X:=F]} A$, which implies $(M', \vec{P}) \in \perp\!\!\!\perp$, which shows $M' \Vdash_\rho \forall X.A$ by the genericity of $\vec{P}$.

Let δ end with

$$\frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash M : A[B/X]}$$

Let $\vec{P} \in \|A[B/X]\|_\rho$. By the Substitution Lemma 41, $\vec{P} \in \|A\|_{\rho[X:=\|B\|_\rho]}$. Now, the induction hypothesis gives us $M' \Vdash_\rho \forall X.A$; but, by definition, this means that $M' \Vdash_{\rho[X:=F]} A$ for all $F \in \mathbb{F}$, in particular when $F = \|B\|_\rho$. Hence, $(M', \vec{P}) \in \perp\!\!\!\perp$, which allows us to conclude. □

The Adequacy Lemma 42 is quite powerful: it tells us that **F**-typable terms lie in the intersection of all admissible sets.

Theorem 43 *If M is* **F***-typable and $\Lambda_0 \subset \Lambda$ is admissible, then $M \in \Lambda_0$.*

PROOF. Let Λ_0 be an admissible set of λ-terms; by definition, there exist a Λ_0-pole and a valuation ρ such that, for every formula A, $\mathcal{V} \subseteq \|A\|_\rho^\perp \subseteq \Lambda_0$. Now, $\Gamma \vdash M : A$ is derivable in "Curry-style" system **F** for some formula A, with Γ containing the free variables of M. Since, under ρ, variables realize any type, we may apply the Adequacy Lemma 42 to the substitution $M[\vec{x}/\vec{x}] = M$ (where $\vec{x}$ are the free variables of M), obtaining $M \Vdash_\rho A$, which, by definition of realizability and by adaptation, gives us $M \in \Lambda_0$. □

6.4 Strong normalization via realizability

By Theorem 43, proving the strong normalization property reduces to showing that the set of strongly normalizable terms, denoted by Λ_{SN}, is admissible.

Lemma 44 *For any λ-terms M and N, $M \to M'$ implies $M[N/x] \to M'[N/x]$.*

PROOF. An easy induction on M, using the fact that, for arbitrary λ-terms K, L, $K[N/x][L[N/x]/y] = K[L/y][N/x]$, provided that $y \notin \mathrm{fv}(N)$, which is also shown by a straightforward induction. □

In the sequel, if $M \in \Lambda$ and $\vec{P} = \langle N_1, \ldots, N_n \rangle \in \Pi$, we abusively use the notation $M\vec{P}$ to mean $MN_1 \cdots N_n$.

Lemma 45 *Let $M \in \Lambda$, $N \in \Lambda_{\mathrm{SN}}$, and $\vec{P} \in \Pi$. Then, $M[N/x]\vec{P} \in \Lambda_{\mathrm{SN}}$ implies $(\lambda x.M)N\vec{P} \in \Lambda_{\mathrm{SN}}$.*

PROOF. For simplicity, we set $M_0 = (\lambda x.M)N\vec{P}$ and $M_0' = M[N/x]\vec{P}$. First of all, we argue that $M_0' \in \Lambda_{\mathrm{SN}}$ implies $M \in \Lambda_{\mathrm{SN}}$ and $\vec{P} \in \Pi_{\mathrm{SN}}$. The obvious fact that any subterm of a strongly normalizable term is itself strongly normalizable immediately gives us $\vec{P} \in \Pi_{\mathrm{SN}}$ and $M[N/x] \in \Lambda_{\mathrm{SN}}$; then, Lemma 44 yields $M \in \Lambda_{\mathrm{SN}}$, because an infinite reduction sequence starting from M would yield an infinite reduction sequence starting from $M[N/x]$.

Suppose now, for the sake of contradiction, that there exists an infinite reduction sequence starting from M_0; this necessarily has the form

$$M_0 \to^* (\lambda x.M')N'\vec{P}' \to M'[N'/x]\vec{P}' \to^* \cdots,$$

where $M \to^* M'$, $N \to^* N'$, and $P_i \to^* P_i'$ for every term of the sequence $\vec{P}$. This is because, if the sequence never reduced the head redex, we would have that one of M, N or one of the terms of $\vec{P}$ is not strongly normalizable, against what we proved above. Now, using again Lemma 44, we have $M[N/x] \to^* M'[N/x]$, so that

$$M_0' \to^* M'[N/x]\vec{P} \to^* M'[N'/x]\vec{P}',$$

which shows that M_0' is not strongly normalizable, against the hypothesis. □

We now consider the set

$$\perp\!\!\!\perp_0 = \{(M, \vec{P}) \in \Lambda \times \Pi \mid M\vec{P} \in \Lambda_{\mathrm{SN}}\},$$

which is easily seen to be a Λ_{SN}-pole, using Lemma 45. In the following, we write Π_{SN} for the set of finite sequences of strongly normalizable λ-terms.

Lemma 46 *Let $F \in \mathbb{F}_{\Lambda_{\mathrm{SN}}}$. Then, with respect to the pole $\perp\!\!\!\perp_0$, $\mathcal{V} \subseteq F^{\perp} \subseteq \Lambda_{\mathrm{SN}}$.*

Proof. Let $x \in \mathcal{V}$ and $\vec{P} \in F$; since $\vec{P} \in \Pi_{\mathrm{SN}}$, obviously $x\vec{P} \in \Lambda_{\mathrm{SN}}$, so $x \in F^{\perp}$.

Let now $M \in F^{\perp}$. Since there is at least one $\vec{P} \in F$, we may consider the λ-term $M\vec{P}$, which is in Λ_{SN} by orthogonality; hence, $M \in \Lambda_{\mathrm{SN}}$, because it is a subterm of a strongly normalizable term. □

Proposition 47 *The set Λ_{SN} is admissible.*

Proof. We shall use the set $\perp\!\!\!\perp_0$ introduced above as our pole, and consider the valuation ρ_0 such that $\rho_0(X) = \Pi_{\mathrm{SN}}$ for all $X \in \mathcal{A}$. We claim that, for every formula A, $\|A\|_{\rho_0} \in \mathbb{F}_{\Lambda_{\mathrm{SN}}}$, which, by Lemma 46, is enough to conclude.

Proving the claim is done by induction on A, with a little bit of "induction loading": in fact, we prove that $\emptyset \subsetneq \|A\|_{\rho_0'} \subseteq \Pi_{\mathrm{SN}}$ holds for every ρ_0' of the form $\rho_0[X_1 := F_1] \cdots [X_n := F_n]$ for some $n \in \mathbb{N}$, some propositional atoms $X_1, \ldots, X_n$, and some $F_1, \ldots, F_n \in \mathbb{F}_{\Lambda_{\mathrm{SN}}}$.

The atomic case is trivial. The case $A = \forall X.A_0$ is immediate (this is where the induction loading is needed). If $A = A_1 \Rightarrow A_2$, we use the induction hypothesis and Lemma 46 to infer that $\|A_1\|_{\rho_0'}^{\perp}$ and $\|A_2\|_{\rho_0'}$ are non-empty, which implies the non-emptiness of $\|A\|_{\rho_0'}$; similarly, by the induction hypothesis and Lemma 46, we have $\|A_1\|_{\rho_0'}^{\perp} \subseteq \Lambda_{\mathrm{SN}}$ and $\|A_2\|_{\rho_0'} \subseteq \Pi_{\mathrm{SN}}$, which implies $\|A\|_{\rho_0'} \subseteq \Pi_{\mathrm{SN}}$. □

Corollary 48 (Strong normalization) *Every* **F***-typable term is strongly normalizable.*

Proof. By Theorem 43, applied to Proposition 47. □

7 Denotational Semantics

7.1 The case of the simply typed λ-calculus

The field of denotational semantics was develop originally for the λ-calculus. The aim is to find a *denotation* for λ-terms, *i.e.*, something which reflects he intuitive meaning of λ-terms, which are supposed to be functions. So, for instance, if we denote by $[\![M]\!]$ the denotation of a λ-term M containing a free variable x, we would like $[\![M]\!]$ to be a function of x and if N is another λ-term, we would like $[\![M[N/x]]\!]$ to denote the result of applying the function $[\![M]\!]$ to the argument $[\![N]\!]$. In particular, denotational semantics must be an *invariant of computation*:

$$M \to M' \qquad \text{implies} \qquad [\![M]\!] = [\![M']\!].$$

Moreover, if the denotation of N and N' is equal, then their image through the function $[\![M]\!]$ should be equal

$$[\![N]\!] = [\![N']\!] \qquad \text{implies} \qquad [\![M[N/x]]\!] = [\![M[N'/x]]\!].$$

For the simply typed λ-calculus, it is not hard to give an interpretation satisfying the above requirements, with λ-terms denoting set-theoretic functions. We start by choosing, for each atomic type X, a set $[\![X]\!]$. Then, we define

$$[\![T \to U]\!] = [\![U]\!]^{[\![T]\!]},$$

where by Y^X we denote the set of all functions from the set X to the set Y. The fundamental remark at this point is that, given any three sets Γ, X, Y there is a bijection

$$\Phi_{\Gamma,X,Y} : Y^{\Gamma \times X} \to (Y^X)^\Gamma,$$

which maps a function $f : \Gamma \times X \to Y$ to its "Curryed" version $\Phi(f) : \Gamma \to Y^X$.

Using the above bijection, we may define the interpretation of λ-terms by induction on their structure. Given a λ-term $M : U$ and a sequence $\rho = x_1^{T_1}, \ldots, x_n^{T_n}$ containing at least the free variables of M, we define

$$[\![M]\!]_\rho : [\![T_1]\!] \times \cdots \times [\![T_n]\!] \to [\![U]\!]$$

as follows (for brevity, we set $\Gamma := [\![T_1]\!] \times \cdots \times [\![T_n]\!]$):

- if $M = x^U$, then we must have $x_i = x$ and $T_i = U$ for some i, and we define $[\![M]\!]_\rho$ to be the projection on the i-th argument;
- if $M = \lambda x^T.N$, then we must have $U = T \to V$ and $N : V$. By induction,

  $$[\![N]\!]_{\rho,x^T} : \Gamma \times [\![T]\!] \to [\![U]\!]$$

  is already defined, and we set

  $$[\![M]\!]_\rho = \Phi_{\Gamma,[\![T]\!],[\![V]\!]}([\![N]\!]_{\rho,x^T}),$$

  which the reader my check to be of the right type.
- If $M = NP$, then we must have $N : T \to U$ and $P : T$. By induction

  $$[\![N]\!]_\rho : \Gamma \to [\![U]\!]^{[\![T]\!]} \qquad \text{and} \qquad [\![P]\!]_\rho : \Gamma \to [\![T]\!]$$

  are already defined. From that, we define $[\![M]\!]_\rho$ to be the function

  $$\begin{array}{rccl} [\![M]\!]_\rho : & \Gamma & \to & [\![U]\!] \\ & c & \mapsto & \Phi^{-1}_{\Gamma,[\![T]\!],[\![U]\!]}([\![N]\!]_\rho)(c, [\![P]\!]_\rho(c)). \end{array}$$

  Again, the reader may check that the expression is well-typed.

Although we shall not verify it here, it is not hard to show that the following holds:

Theorem 49 *For any interpretation of atomic types, for every simply typed λ-terms M, N and for every ρ containing the free variables of M and N,*

$$M \simeq_{\beta\eta} N \qquad \textit{implies} \qquad [\![M]\!]_\rho = [\![N]\!]_\rho.$$

By inspecting of the definition of interpretation, we realize that the only essential point is to be able to "Curry" and "un-Curry" functions. This is possible in any *Cartesian closed category*, *i.e.*, a category $\mathcal{C}$ with finite products and such that, for any objects X, Y, there is an object Y^X such that, for any other object Γ, we have an isomorphism

$$\mathcal{C}[\Gamma \times X, Y] \cong \mathcal{C}[\Gamma, Y^X]$$

which is natural in Γ, X and Y. We invite the reader to check that the above definitions may be rephrased, almost word by word, in any Cartesian closed category: we simply replace the word "set" by "object", "function" by "morphism", and Φ is the natural isomorphism just described. The naturality of Φ is needed for Theorem 49 to hold in the general case.

7.2 The pure λ-calculus

Set-theoretic models of the simply-typed λ-calculus like the one above have been known for a long time, at least from the 50s and 60s. However, until the mid-60s, there was an almost universal consensus that the computationally much more interesting *pure* λ-calculus could not have any model. This is certainly true from the set-theoretic point of view: a model of the pure λ-calculus like the one above would require a set D such that D^D may be injected in D, a possibility ruled out long ago by Cantor.

In the late 60s, Dana Scott came up with the idea of solving the stalemate by using something more than just plain sets. Retrospectively, this may be justified by looking at a situation which is very well-known in analysis: although the set of all functions on the real numbers $\mathbb{R}$ has cardinality strictly greater than the continuum, this is not true if we restrict to *continuous* functions. In fact, let $\mathfrak{c}$ be the cardinality of the continuum, and let $\mathfrak{c}'$ be the cardinality of the set $C^0(\mathbb{R})$ of continuous functions on $\mathbb{R}$.

- Since every constant function is continuous, we have $\mathfrak{c} \leq \mathfrak{c}'$;
- since the set of rational numbers $\mathbb{Q}$, whose cardinality is well-known to be $\aleph_0$, is dense in $\mathbb{R}$, every function in $C^0(\mathbb{R})$ is uniquely determined by the values it takes on rational numbers. Therefore, every function in $C^0(\mathbb{R})$ induces one (and one only) function in $\mathbb{R}^{\mathbb{Q}}$, so we have

$$\mathfrak{c}' \leq \mathfrak{c}^{\aleph_0} = \mathfrak{c}.$$

We conclude that there are as many continuous functions on $\mathbb{R}$ as there are real numbers.

Probably the simplest example of how the above idea may be exploited for yielding models of the pure λ-calculus is *Plotkin's model*. This is based on a general construction which, of course, is due to Scott. We shall give the concrete model first, and the abstract definitions later.

Consider the set $\mathcal{P}(\mathbb{N})$ of all subsets of $\mathbb{N}$, and let $\mathcal{P}_{\mathrm{f}}(\mathbb{N})$ be set of all *finite* subsets of $\mathbb{N}$. Let $e \in \mathcal{P}_{\mathrm{f}}(\mathbb{N})$ and define

$$O_e := \{x \subseteq \mathbb{N} \mid e \subseteq x\}.$$

Obviously the sets O_e, for e spanning $\mathcal{P}_{\mathrm{f}}(\mathbb{N})$, cover $\mathcal{P}(\mathbb{N})$; moreover, for all $e_1, e_2 \in \mathcal{P}_{\mathrm{f}}(\mathbb{N})$, $e_1 \cup e_2$ is obviously still finite and $O_{e_1} \cap O_{e_2} = O_{e_1 \cup e_2}$. Therefore,

the family $\{O_e \mid e \in \mathcal{P}_{\mathrm{f}}(\mathbb{N})\}$ is a basis for a topology on $\mathcal{P}(\mathbb{N})$, which we call the *Scott topology*. It is possible to prove (we shall do this in the general case) the following:

Proposition 50 *A function $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$ is continuous with respect to the Scott topology iff, for ever $x \subseteq \mathbb{N}$, we have*

$$f(x) = \bigcup_{e \subseteq_{\mathrm{f}} x} f(e),$$

where $e \subseteq_{\mathrm{f}} x$ means that e is a finite subset of x.

In other words, Scott-continuous functions are uniquely determined by the values they take on finite sets. Since the set $\mathcal{P}_{\mathrm{f}}(\mathbb{N})$ is denumerable, we recreated a very similar situation to the one described above using real numbers. We denote by $C^s(\mathcal{P}(\mathbb{N}))$ the set of Scott-continuous functions on $\mathcal{P}(\mathbb{N})$.

Thanks to Proposition 50, we may represent every $f \in C^s(\mathcal{P}(\mathbb{N})$ by its *trace*

$$\mathrm{tr}(f) := \{(e, n) \in \mathcal{P}_{\mathrm{f}}(\mathbb{N}) \times \mathbb{N} \mid n \in f(e)\}.$$

Note that the trace is not the graph of f restricted to finite arguments (otherwise it would be a subset of $\mathcal{P}_{\mathrm{f}}(\mathbb{N}) \times \mathcal{P}(\mathbb{N})$, whereas it is a subset of $\mathcal{P}_{\mathrm{f}}(\mathbb{N}) \times \mathbb{N}$). If $f(e) = y$, the trace contains a pair (e, n) for each $n \in y$ and there may be infinitely many such pairs, because y need not be finite, even if e is.

This "trick" of considering the trace of f instead of its graph is essential to inject $C^s(\mathcal{P}(\mathbb{N}))$ into $\mathcal{P}(\mathbb{N})$ itself. For this, we fix two bijections

$$\begin{aligned} \ulcorner \cdot \urcorner &: \mathcal{P}_{\mathrm{f}}(\mathbb{N}) \to \mathbb{N}, \\ \beta &: \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \end{aligned}$$

and define, given a Scott-continuous function $f : \mathcal{P}(\mathbb{N}) \to \mathcal{P}(\mathbb{N})$, the set

$$\Phi(f) := \{\beta(\ulcorner e \urcorner, n) \mid (e, n) \in \mathrm{tr}(f)\}.$$

It is immediate to check that Φ is an injection $C^s(\mathcal{P}(\mathbb{N})) \hookrightarrow \mathcal{P}(\mathbb{N})$.

Conversely, we may define a map $\Psi : \mathcal{P}(\mathbb{N}) \twoheadrightarrow C^s(\mathcal{P}(\mathbb{N}))$ as follows. If $\llcorner \cdot \lrcorner$ is the inverse of $\ulcorner \cdot \urcorner$ and if $\pi_1, \pi_2 : \mathbb{N} \to \mathbb{N}$ are the two "projections" such that $\beta^{-1} = (\pi_1, \pi_2)$, given $u \subseteq \mathbb{N}$ we define the function

$$\begin{array}{rccl} \Psi_0(u) : & \mathcal{P}_{\mathrm{f}}(\mathbb{N}) & \to & \mathcal{P}(\mathbb{N}) \\ & e & \mapsto & \{\pi_2(n) \mid n \in u,\ \llcorner \pi_1(n) \lrcorner = e\}. \end{array}$$

One may check that, for all $u \in \mathcal{P}(\mathbb{N})$, $\Psi_0(u)$ is continuous. Therefore, since $\mathcal{P}_{\mathrm{f}}(\mathbb{N})$ is dense in $\mathcal{P}(\mathbb{N})$, it uniquely extends to a continuous function $\Psi(u)$, which is defined by

$$\Psi(u)(x) = \{\pi_2(n) \mid n \in u,\ \llcorner \pi_1(n) \lrcorner \subseteq x\},$$

for all $x \in \mathcal{P}(\mathbb{N})$.

It it easy to verify that $\Psi \circ \Phi$ is the identity on $C^s(\mathcal{P}(\mathbb{N}))$. The converse is not true: Ψ is not injective because there is more than one way of specifying a continuous function by a "trace", *i.e.*, every subset $t \subseteq \mathcal{P}_{\mathrm{f}}(\mathbb{N}) \times \mathbb{N}$ induces a continuous function $\mathrm{fun}(t)$ by setting

$$\mathrm{fun}(t)(x) = \{n \mid (e, n) \in u,\ e \subseteq x\}, \quad \forall x \in \mathcal{P}(\mathbb{N}),$$

but different subsets of $\mathcal{P}_{\mathrm{f}}(\mathbb{N}) \times \mathbb{N}$ may induce the same function. For instance, if $f \in C^s(\mathcal{P}(\mathbb{N}))$ and $(e,n),(e',n) \in \mathrm{tr}(f)$ with $e \subseteq e'$, we invite the reader to check that, if we set $t = \mathrm{tr}(f) \setminus \{(e',n)\}$, we still have $\mathrm{fun}(t) = f$.

Nevertheless, the presence of the two maps Φ and Ψ, which are called a *retraction pair*, is enough to adapt the definitions of Sect. 7.1 to the untyped setting: a pure λ-term M whose free variables are contained in the list $\rho = x_1, \ldots, x_n$ will be interpreted by a continuous function

$$[\![M]\!]_\rho : \overbrace{\mathcal{P}(\mathbb{N}) \times \cdots \times \mathcal{P}(\mathbb{N})}^{n} \to \mathcal{P}(\mathbb{N})$$

(contrarily to the familiar situation in analysis, Scott-continuity on the product is equivalent to separate Scott-continuity in each argument), and whenever we use Φ, Φ^{-1} in the simply-typed λ-calculus, we use Φ, Ψ in Plotkin's model. More importantly, we have

Theorem 51 (Plotkin) *For every pure λ-terms M, N and for every sequence of variables ρ containing the free variables of M, N, if we denote by $[\![\cdot]\!]_\rho$ the interpretation in Plotkin's model, we have*

$$M \simeq_\beta N \quad \textit{implies} \quad [\![M]\!]_\rho = [\![N]\!]_\rho.$$

We observe that, contrarily to Theorem 49, η-equivalence is not validated by Plotkin's model. However, Plotkin exhibited a particular choice of the bijections $\ulcorner \cdot \urcorner : \mathcal{P}_{\mathrm{f}}(\mathbb{N}) \to \mathbb{N}$ and $\beta : \mathbb{N}^2 \to \mathbb{N}$ used in the definition of the model such that, for all ρ as above,

$$[\![M]\!]_\rho = [\![N]\!]_\rho \quad \text{iff} \quad \mathfrak{B}(M) = \mathfrak{B}(N),$$

where $\mathfrak{B}(P)$ denotes the Böhm tree of the pure λ-term P. The problem of characterizing the equality of a pure λ-calculus model (the so-called *theory* of the model) is a very interesting one, but we shall not speak more about this here.

As in the simply-typed case, the situation we have in Plotkin's model is an instance of a more general pattern: a denotational model of the pure λ-calculus is found as soon as we have a *reflexive object* in a Cartesian closed category. By reflexive object we mean an object D with morphisms

$$D^D \underset{\Psi}{\overset{\Phi}{\rightleftarrows}} D$$

such that $\Psi \circ \Phi = id_{D^D}$. If we actually have $\Psi = \Phi^{-1}$, then the model will be *extensional*, *i.e.*, it will also validate η-equivalence.

So $\mathcal{P}(\mathbb{N})$ should be a reflexive object in a Cartesian closed category... but which one? In the next section we shall see that the Scott topology we defined on $\mathcal{P}(\mathbb{N})$ is actually a special case of a more general definition, yielding a so-called *Scott domain*. Scott domains and Scott-continuous functions form a Cartesian closed category, of which $\mathcal{P}(\mathbb{N})$ is a reflexive object. Indeed, although we didn't mention this because we cannot yet define what it means, a fundamental fact in the construction on Plotkin's model is that Φ and Ψ are themselves Scott-continuous, *i.e.*, they are morphisms of the category.

A posteriori, as the reader will have remarked at this point, Scott's essential achievement was to find a Cartesian closed category in which reflexive objects exist. The belief, widespread at the time, that the pure λ-calculus had no models is justified by the fact that people were only considering the prototypical Cartesian closed category, that of sets and functions, which has no reflexive objects.

7.3 Scott domains and continuous functions

Let us give some preliminary definitions. If $(D, \leq)$ is a poset (partially ordered set), and if $x, y \in D$, we denote by $x \vee y$ (resp. $x \wedge y$) the supremum (resp. infimum) of x and y, if it exists. Similarly, if $A \subseteq D$, we denote by $\bigvee A$ the supremum of all the elements of A, if it exists. We write $x \uparrow y$ if there exists z such that $x \leq z$ and $y \leq z$. We denote by $\downarrow x$ the principal ideal of the element x, *i.e.*

$$\downarrow x := \{y \in D \mid y \leq x\}.$$

Definition 12 (Directed set, Scott topology) *Let $(D, \leq)$ be a partial order. A set $\Delta \subseteq D$ is* directed *if, for all $x, y \in \Delta$, there exists $z \in \Delta$ such that $x \leq z$ and $y \leq z$.*

The Scott topology *on D is determined as follows: a set $O \subseteq D$ is open if*

1. *it is* upward-closed*, i.e., $x \in O$ and $x \leq x'$ implies $x' \in O$;*
2. *it is* inaccessible by directed suprema*: for every directed $\Delta \subseteq D$ such that $\bigvee \Delta$ exists, $\bigvee \Delta \in O$ implies $\Delta \cap O \neq \emptyset$.*

The following gives important examples of Scott-open sets:

Lemma 52 *Let $(D, \leq)$ be a poset. Then, for all $x \in D$, the set*

$$x^\circ := \{y \in D \mid y \not\leq x\} = \complement \downarrow x$$

is open in the Scott topology on D.

Proof. The set x° is clearly upward-closed. Let Δ be directed such that its supremum exists, and suppose that $\bigvee \Delta \in x^\circ$, *i.e.*, $\bigvee \Delta \not\leq x$. If $\Delta \cap x^\circ = \emptyset$, we would have $\Delta \subseteq \downarrow x$, so x is an upper bound of Δ, so $\bigvee \Delta \leq x$, contrarily to the hypothesis. □

If the posets involved are *complete*, continuous functions with respect to their Scott topologies may actually be defined in a purely order-theoretic way, without the need to invoke any topology at all.

Definition 13 (Complete partial order (cpo)) *A poset $(D, \leq)$ is* complete *if it has a least element, denoted by $\bot$, and if the supremum of every directed set exists.*

Proposition 55 below gives a purely order-theoretic characterization of continuous functions between cpos endowed with their Scott topology. To prove it, we first need a couple of preliminary topological definitions and results.

Given a topological space X, we define its *specialization preorder* $\preceq$ as the following relation on X: $x \preceq x'$ just if for every open set O, $x \in O$ implies $x' \in O$.

Lemma 53 *Every continuous function between two topological spaces X, Y is monotonic with respect to their specialization preorders.*

PROOF. Let $f : X \to Y$ be continuous. We shall prove its monotonicity by contraposition. Let $x, x' \in X$ be such that $f(x) \not\preceq f(x')$ in Y. This means that there exists an open set $O \subseteq Y$ such that $f(x) \in O$ but $f(x') \notin O$. By continuity, $f^{-1}(O)$ is an open set of X, and obviously $x \in f^{-1}(O)$ whereas $x' \notin f^{-1}(O)$, so $x \not\preceq x'$ in X. □

Lemma 54 *The specialization preorder of the Scott topology on a poset $(D, \leq)$ coincides with $\leq$.*

PROOF. The fact that $x \leq x'$ implies $x \preceq x'$ is an immediate consequence of the fact that Scott-open sets are upward-closed w.r.t. $\leq$. Let now $x \preceq x'$, and suppose, for the sake of contradiction, that $x \not\leq x'$. This means $x \in x'^{\circ}$, but then, by Lemma 52 and the definition of $\preceq$, we would have $x' \in x'^{\circ}$ and therefore $x' \not\leq x'$, contradicting reflexivity. □

Proposition 55 *Let $(D, \leq)$, $(E, \leq')$ be cpos, and let $f : D \to E$. Then, f is continuous with respect to the Scott topologies on D and E iff for every directed $\Delta \subseteq D$ such that $\bigvee \Delta$ exists, we have:*

1. *the set $f(\Delta) = \{f(x) \mid x \in \Delta\}$ is also directed;*
2. *$f(\bigvee \Delta) = \bigvee f(\Delta)$.*

PROOF. Let f be topologically continuous. By Lemmas 53 and 54, f is monotonic with respect to $\leq$ and $\leq'$, from which one easily infers the inequality $\bigvee f(\Delta) \leq' f(\bigvee \Delta)$, as well as property 1 (monotonic functions map directed sets to directed sets). To complete the proof of property 2, suppose that $\bigvee f(\Delta) <' f(\bigvee \Delta)$, *i.e.*, suppose that $f(\bigvee \Delta) \in (\bigvee f(\Delta))^{\circ}$. By continuity, the set $O = f^{-1}((\bigvee f(\Delta))^{\circ})$ is Scott-open, and by definition $\bigvee \Delta \in O$ implies that there exists $x \in \Delta$ such that $x \in O$, which in turn implies that $f(x) \in (\bigvee f(\Delta))^{\circ}$, which is impossible, because we know that $f(x) \leq' \bigvee f(\Delta)$ since $x \in \Delta$.

Suppose now that f verifies conditions 1 and 2, take a Scott-open set $O \subseteq E$, a directed set $\Delta \subseteq D$, and assume that $\bigvee \Delta \in f^{-1}(O)$. This implies that $f(\bigvee \Delta) = \bigvee f(\Delta) \in O$, which by definition of Scott-open implies that $f(x) \in O$ for some $x \in \Delta$, which means that $x \in \Delta \cap f^{-1}(O)$, proving that $f^{-1}(O)$ is also Scott-open and that f is continuous. □

Hence, when working with cpos (as is always the case in denotational semantics), properties 1 and 2 of Proposition 55 may be taken as the definition of Scott-continuous function and there is no need to introduce the Scott topology.

Furthermore, one usually considers cpos having additional properties which make continuity enjoy even more concrete (and intuitive) equivalent formulations, such as the one of Proposition 50 in Plotkin's model. This is what we shall do next.

Intuitively, one must think of the elements of a cpo as "pieces of information", the least element $\bot$ being the absence of information. With this view in mind, the relation $x \leq y$ may be read as "the information x approximates the information y". The following definition is an abstract way of speaking of "finite information".

Definition 14 (Compact element) *Let $(D, \leq)$ be a poset. An element $d \in D$ is* compact *if for all directed $\Delta \subseteq D$ such that $\bigvee \Delta$ exists, $d \leq \bigvee \Delta$ implies that there exists $x \in \Delta$ such that $d \leq x$.*

We denote by $\mathcal{K}(D)$ the set of compact elements of a poset D, and given $x \in D$, we set $\mathcal{K}(x) = \{d \in \mathcal{K}(D) \mid d \leq x\}$.

Intuitively, $\mathcal{K}(x)$ is the set of finite approximations of x. The fact that compact elements correspond to finite information is supported by the characterization (2) of continuous functions given in Proposition 56 below: a continuous function is entirely determined by the values it takes on compact elements, just as a continuous function on real numbers (which may contain infinite information) is entirely determined by the values it takes on the rational numbers (which contain finite information).

Topologically speaking, a compact element is similar to an isolated point (in fact, some people call compact elements *isolated*): in Definition 14 the special case $\bigvee \Delta = d$ gives us necessarily $d \in \Delta$. Therefore, just like the only sequences converging to an isolated point are those which are eventually equal to that point, the only directed sets whose supremum is a compact element are those already containing that element. In general, the analogy "supremum = limit" is very useful when dealing with Scott topologies.

Let $f : D \to E$ be a monotonic function between posets. Let $x \in D$ and $e \in \mathcal{K}(f(x))$; we define

$$\mathrm{apx}_f(x, e) := \{d \in \mathcal{K}(x) \mid e \leq f(d)\}.$$

The idea of the above definition is that, given a finite approximation e of $f(x)$, $d \in \mathrm{apx}_f(x, e)$ is a finite approximation of x such that $f(d)$ approximates $f(x)$ at least as well as e. This is another intuition behind continuity, formalized by characterization (3) of Proposition 56: to obtain a finite approximation of the output, a continuous function only needs a finite approximation of the input.

Definition 15 (Scott domain) *A* Scott domain *is a cpo $(D, \leq)$ which is further:*

bounded complete: *for all $x, y \in D$, $x \uparrow y$ implies that $x \vee y$ exists;*

algebraic: *for all $x \in D$, $\bigvee \mathcal{K}(x) = x$.*

The essential property of Scott domains is that they are algebraic. Bounded completeness is a secondary property; it implies that the set $\mathcal{K}(x)$ is directed, so that its supremum exists (indeed, we invite the reader to prove that if d, d' are compact and $d \vee d'$ exists, then it is compact as well; then, if $d, d' \in \mathcal{K}(x)$, obviously $d \uparrow d'$, and by bounded completeness and the above remark we have $d \vee d' \in \mathcal{K}(x)$).

As anticipated above, continuous functions on Scott domains may be characterized as follows:

Proposition 56 *Let D, E be Scott domains, and let $f : D \to E$. Then, the following are equivalent:*

1. *f is continuous;*
2. *for every $x \in D$, $f(x) = \bigvee_{d \in \mathcal{K}(x)} f(d)$;*

3. f is monotonic and for all $x \in D$ and $e \in \mathcal{K}(f(x))$, $\mathrm{apx}_f(x,e) \neq \emptyset$.

PROOF. That (1) implies (2) is an immediate consequence of the fact that D is algebraic, via continuity:

$$f(x) = f\left(\bigvee_{d \in \mathcal{K}(x)} d\right) = \bigvee_{d \in \mathcal{K}(x)} f(d).$$

Let us prove that (2) implies (3). Monotonicity is a consequence of the obvious fact that $x \leq x'$ implies $\mathcal{K}(x) \subseteq \mathcal{K}(x')$, and so $f(x) = \bigvee_{d \in \mathcal{K}(x)} f(d) \leq \bigvee_{d' \in \mathcal{K}(x')} f(d') = f(x')$. For what concerns the other part of property (3), if $x \in D$ and $e \in \mathcal{K}(f(x))$, we have $e \leq f(x) = \bigvee_{d \in \mathcal{K}(x)} f(d)$, so by compactness of e we must have $e \leq f(d)$ form some $d \in \mathcal{K}(x)$, which means by definition that $d \in \mathrm{apx}_f(x,e)$.

Let us now consider the implication of (3) to (1). Let $\Delta \subseteq D$ be directed. Monotonicity immediately gives us that $f(\Delta)$ is also directed, as well as the inequality $\bigvee f(\Delta) \leq f(\bigvee \Delta)$. This inequality implies in particular that $\mathcal{K}(\bigvee f(\Delta)) \subseteq \mathcal{K}(f(\bigvee \Delta))$. We shall prove the reverse inclusion, which allows us to conclude thanks to the fact that E is algebraic (if $y, y' \in E$ are such that $\mathcal{K}(y) = \mathcal{K}(y')$, then necessarily $y = \bigvee \mathcal{K}(y) = \bigvee \mathcal{K}(y') = y'$). So let $e \in \mathcal{K}(f(\bigvee \Delta))$. We know that there is at least one $d \in \mathrm{apx}_f(\bigvee \Delta, e)$, which by definition is such that $d \in \mathcal{K}(\bigvee \Delta)$ and $e \leq f(d)$. But, by compactness of d, $d \leq \bigvee \Delta$ implies $d \leq x$ for some $x \in \Delta$, which by monotonicity implies $e \leq f(d) \leq f(x) \leq \bigvee f(\Delta)$, which proves $e \in \mathcal{K}(\bigvee f(\Delta))$, as desired. □

7.4 Berry's dI-domains and stable functions

In order to capture the notion of *sequential* computation, which is the one expressed by the λ-calculus, in the early 80s Gérard Berry introduced a refinement of Scott domains, which have more structure and satisfy one further property.

Definition 16 (dI-domain) *A* dI-domain *is a Scott domain $(D, \leq)$ in which all binary infima exist and which satisfies the following:*

property d (distributivity): *if $x, y, z \in D$ are such that there exists $w \in D$ such that $x, y, z \leq w$, then $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$;*

property I: *for all $d \in \mathcal{K}(D)$, $\mathcal{K}(d)$ is finite.*

Distributivity is a useful property, but we shall not use it here. Property I is the essential one: it states that compact points really are "finite". In fact, it implies the following:

Lemma 57 *Let $(D, \leq)$ be a dI-domain. Then:*

1. *for all $x \in D$, $x \in \mathcal{K}(D)$ iff $\downarrow x$ is finite;*
2. *as a consequence, if $d, e \in \mathcal{K}(D)$, then $d \wedge e \in \mathcal{K}(D)$.*

PROOF. Let us start with point 1. The forward implication is shown by an easy induction on the cardinality of $\mathcal{K}(x)$ (which is finite because of property I). The only interesting case is the base case $\mathcal{K}(d) = \{d\}$, with $d \neq \bot$, where we use the fact that D is algebraic: if there were any $x < d$, we would have to have $x = \bigvee \mathcal{K}(x) = \bigvee \emptyset = \bot$; but $\bot$ is always compact, so $\mathcal{K}(d) = \{\bot, d\}$, contradiction. The backward implication of point 1 is true in any Scott domain: if $\downarrow x$ is finite, $\mathcal{K}(x)$ must be finite too; but D is algebraic, so $\mathcal{K}(x)$ is directed, and the only finite directed sets are those with a greatest element. Since $\bigvee \mathcal{K}(x) = x$, we conclude that $x \in \mathcal{K}(x)$, so it is compact.

Point 2 is an immediate consequence of point 1: $d, e \in \mathcal{K}(D)$ implies that $\downarrow d$ and $\downarrow e$ are both finite; but then $\downarrow(d \wedge e) = \downarrow d \cap \downarrow e$, is also finite, so $\mathcal{K}(d \wedge e)$ is finite and $d \wedge e \in \mathcal{K}(D)$ as well. □

Consider $D = \mathbb{N} \cup \{\omega, d, e\}$, with the usual order $\leq$ on $\mathbb{N}$ plus the relations $n < \omega$ for all $n \in \mathbb{N}$ and $\omega < d$, $\omega < e$. We invite the reader to check that $(D, \leq)$ is a Scott domain (with distributive infima) which is not a dI-domain, which furthermore gives a counterexample to Lemma 57: d and e are both compact, but $\downarrow d$ and $\downarrow e$ are both infinite and $d \wedge e = \omega$ is not compact.

The notion of function capturing sequentiality, which we mentioned above, is the following:

Definition 17 (Stable function) *Let D, E be dI-domains. A function $f : D \to E$ is* stable *if it is continuous and if, furthermore, for all $x, x' \in D$,*

$$x \uparrow x' \quad \textit{implies} \quad f(x \wedge x') = f(x) \wedge f(x').$$

For stable functions, Proposition 56 may be strengthened as follows:

Proposition 58 *Let D, E be dI-domains, and let $f : D \to E$. Then, f is stable iff it is monotonic and, for all $x \in D$ and $e \in \mathcal{K}(f(x))$, $\mathrm{apx}_f(x, e)$ has a least element.*

PROOF. Suppose that f is stable, and take $x \in D$, $e \in \mathcal{K}(f(x))$. By definition, f is continuous and hence monotonic. Moreover, we know that $\mathrm{apx}_f(x, e)$ is a non-empty subset of $\mathcal{K}(D)$. Since D is a dI-domain, $\mathrm{apx}_f(x, e)$ necessarily has minimal elements (property I guarantees that there are no infinitely descending chains of compact elements). Choose one of them, let us call it d, and suppose, for the sake of contradiction, that there exists $d' \in \mathrm{apx}_f(x, e)$ such that $d \not\leq d'$. Note that $d \uparrow d'$ because both $d, d' \leq x$. Now, since $e \leq f(d), f(d')$, we have, by stability, $e \leq f(d) \wedge f(d') = f(d \wedge d')$. But by point 2 of Lemma 57 $d \wedge d' \in \mathcal{K}(D)$, so $d \wedge d' \in \mathrm{apx}_f(x, e)$, which contradicts the minimality of d, because $d \wedge d' < d$.

Suppose now that f is as stated. Point 3 of Proposition 56 already assures us that f is continuous, so let $x \uparrow y \in D$. Since $x \wedge y \leq x, y$, we immediately have $f(x \wedge y) \leq f(x) \wedge f(y)$. We will prove the converse inequality by showing that $\mathcal{K}(f(x) \wedge f(y)) \subseteq \mathcal{K}(f(x \wedge y))$, which is enough to conclude by algebraicity. Let $e \in \mathcal{K}(E)$ be such that $e \leq f(x) \wedge f(y)$. If $z \in D$ is such that $x, y \leq z$ (which exists because $x \uparrow y$), by monotonicity we have $e \leq f(x), f(y), f(z)$. So let

$$\begin{aligned} d_1 &= \min \mathrm{apx}_f(x, e), \\ d_2 &= \min \mathrm{apx}_f(y, e), \\ d &= \min \mathrm{apx}_f(z, e), \end{aligned}$$

which all exist by hypothesis. Observe now that $d_1, d_2 \in \mathrm{apx}_f(z, e)$, so $d \leq d_1, d_2$, but then we have $d \in \mathrm{apx}_f(x, e) \cap \mathrm{apx}_f(y, e)$, which implies that $d_1 = d_2 = d$. Since $d \leq x, y$, we have $d \leq x \wedge y$, so by monotonicity $e \leq f(d) \leq f(x \wedge y)$, as desired. □

So a stable function f has, with respect to a continuous one, the following additional property: when we want to compute $f(x)$ up to a finite approximation e, there is a *smallest* finite approximation of x which suffices. In the case of a continuous function, we may have several (even infinitely many!) approximations of x doing the job, none standing out as canonical.

Coherence spaces are a special case of dI-domains. They were introduced by Girard with the intent of giving a denotational semantics of system **F**. A deeper analysis of coherence spaces led to the introduction of *linear functions*, a further refinement of stable functions, which successively gave birth to linear logic. We refer the reader to [GLT89] for an exhaustive treatment of coherence spaces, linear functions and their relation with linear logic.

References

[Gir87] Jean-Yves Girard. *Proof Theory and Logical Complexity*, volume 1. Bibliopolis, 1987.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[Kri93] Jean-Louis Krivine. *Lambda-calculus, types and models*. Ellis Horwood, 1993.

[Kri04] Jean-Louis Krivine. Realizability in classical logic, 2004. Lecture notes available at `http://www.pps.jussieu.fr/~krivine/`.

[Tai67] William W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–212, 1967.

[Wel99] Joe B. Wells. Typability and type checking in system F are equivalent and undecidable. *Annals of Pure and Applied Logic*, 98(1–3):111–156, 1999.