Proofs and Programs

Course Notes

Colin Riba
ENS Lyon, Université de Lyon, LIP[*]
`colin.riba@ens-lyon.fr`

March 27, 2023

The main reference for these notes is the book [SU06]. However, we frequently depart from the latter regarding presentation and notation, and also on some technical choices. Other references are given within the text.

[*]UMR 5668 CNRS ENS Lyon UCBL INRIA

Contents

1. Introduction

The main objective of this course is to provide some basics on parts of the theory underlying the Coqproof assistant,[1] namely the **Curry-Howard correspondence** between **proofs** and **programs**:

Prog. Languages		Logic
Types	$\equiv$	Formulae
Programs	$\equiv$	Proofs

The underlying logic of the Coq system, as well as the core of the Curry-Howard correspondence is **intuitionistic** (or **constructive**[2]). Let us try to explain what does this mean. Consider the following simple fact:

Fact 1.1. *Either $(\pi + e)$ or $(\pi - e)$ is irrational.*

Proof. **Assume toward a contradiction** that $(\pi + e)$ and $(\pi - e)$ are both rational. Then $2\pi = (\pi + e) + (\pi - e)$ is rational, a contradiction. □

However, the following questions are open:[3]

- is $(\pi + e)$ irrational?
- is $(\pi - e)$ irrational?

The guiding principles of intuitionistic reasoning are:

- from a proof of a disjunctive statement, say $A \vee B$, one should be able to **extract** either a proof of A or a proof of B; and
- from a proof of a statement of the form $(\forall x \in \mathbb{N})(\exists y \in \mathbb{N})A(x, y)$, one should be able to **extract** a **computable function** $f : \mathbb{N} \to \mathbb{N}$ such that $(\forall x \in \mathbb{N})A(x, f(x))$.

The argument we gave for Fact 1.1 obviously falls short of the first criterion above. This is crucially due to the use of the non-constructive reasoning principle of **reductio ad absurdum** (RAA). Roughly speaking, intuitionistic logic (IL) is defined from classical logic (CL) by

$$(\text{CL}) \quad = \quad (\text{IL}) + (\text{RAA})$$

Intuitionistic (or constructive) mathematics emerged progressively from the 19th century, until their principles were formulated by Brouwer in the early 20th century. It is mainly Heyting who (starting from the 30's) translated Brouwer's principles in logical terms, thus establishing them as mathematical objects.

[1] https://coq.inria.fr/

[2] In this course, we use "intuitionistic" as a technical term qualifying some logical systems, while we use the word "constructive" only informally.

[3] As of the 9th of Jan. 2023, see https://en.wikipedia.org/wiki/Transcendental_number.

Our approach to intuitionistic logic in this course is to insist on **proofs** rather than on **truth** of formulae. In other words, the important mathematical objects are proofs, which (by the Curry-Howard correspondence) will be (mostly) seen as terms of some **typed λ-calculus**. To summarize, our first main topic will be to investigate intuitionistic logics and their relations to typed λ-calculi.

Our second main topic will be to compare intuitionistic systems with their classical counterparts. At first sight, intuitionistic systems are blatantly more restrictive than classical ones since they accept less reasoning principles.

However, there are important logical systems and families of statements for which classical logic is no more powerful than intuitionistic logic. This is crucially the case of formulae expressing program termination. Formulae expressing termination are typically formulae of the form $(\forall x \in \mathbb{N})(\exists y \in \mathbb{N})(\mathtt{t}(x, y) = 0)$ where $\mathtt{t}(x, y)$ represents some primitive recursive function. The latter are called **Π^0_2-formulae**. In many important logical systems containing arithmetic, classical reasoning does not prove more Π^0_2-formulae than intuitionistic reasoning. In other words, in such systems one can prove the termination of the same programs with classical and intuitionistic reasonings.

Moreover (and this is one of the main messages of this course), we see intuitionistic systems as being **richer** than classical ones. Intuitively, discarding (RAA) makes (most) formulae A **not** (logically) equivalent to their double-negation $\neg\neg A$. As a consequence, there are (intuitively) two notions of disjunction in intuitionistic logic:

(1) the plain intuitionistic disjunction $(A \vee B)$,

(2) the "classical" intuitionistic disjunction $\neg\neg(A \vee B)$.

For instance, $(A \vee \neg A)$ is in general **not** intuitionistically provable, but $\neg\neg(A \vee \neg A)$ **is always** intuitionistically provable.

Similarly, there are two existential quantifiers in intuitionistic logic: the plain intuitionistic one $(\exists x)A$ and the "classical" one $\neg\neg(\exists x)A$.

References. The main reference for this course is the book [SU06].

Concerning intuitionistic logic, a good historical account is given in [TvD88a, Chap. 1] (see also the Notes of [SU06, Chap. 2]). As for general technical introductions, besides [SU06, Chap. 2], the textbook [vD04] is very accessible, while [TvD88a, TvD88b] form a more comprehensive account. Other interesting sources include [Bee85, Koh08].

Regarding typed λ-caluli, (besides [SU06]) a good synthetic but general account is [Bar92], while [Pie02] is a gentle but comprehensive reference book from the point of view of programming languages. An important complementary topic to this course is the **denotational semantics** of programming languages, for which we refer to [AC98].

Beware however that we should consistently follow the notations and technical definitions of **none** of these sources.

2. Propositional Logic

This §2 loosely follows [SU06, §2] (as well as parts of [SU06, §6]). Additional references are given within the text.

We consider formulae given by the grammar:

$$A, B \quad ::= \quad \mathsf{p} \quad | \quad A \Rightarrow B \quad | \quad A \wedge B \quad | \quad A \vee B \quad | \quad \top \quad | \quad \bot$$

where p ranges over some (unspecified) set of **atomic propositions**.

Note that there is no primitive notion of negation in the above grammar.

Notation 2.1 (Negation)**.** *We write* $\neg A$ *for* $A \Rightarrow \bot$*.*

We follow the same notational conventions as [SU06, 2.1.2].

Notation 2.2.

(1) We assume that implication ($\Rightarrow$) *associates to the right, so that e.g.* $A \Rightarrow B \Rightarrow C$ *stands for* $A \Rightarrow (B \Rightarrow C)$ *(which is* ***different*** *from* $(A \Rightarrow B) \Rightarrow C$*).*

(2) We assume that negation has the highest priority and implication the lowest, so that e.g. $A \wedge \neg B \Rightarrow C$ *stands for* $(A \wedge (\neg B)) \Rightarrow C$*.*

Finally:

Notation 2.3. *We write* $A \Leftrightarrow B$ *for* $(A \Rightarrow B) \wedge (B \Rightarrow A)$*.*

2.1. Natural Deduction for Intuitionistic Propositional Logic

A **sequent** (for intuitionistic natural deduction) is a pair of the form $\Delta \vdash A$ where A is a formula and Δ is a (possibly empty) **list** of formulae. In the literature, Δ is sometimes called the **antecedent** of $\Delta \vdash A$ and A its **succedent**. We shall often call Δ the **context** of $\Delta \vdash A$. The intended meaning of the sequent $A_1, \ldots, A_n \vdash A$ is $(A_1 \wedge \cdots \wedge A_n) \Rightarrow A$ (see §2.1.1 below).

The system NJ_0 of **natural deduction** for **intuitionistic propositional** logic is given by the deduction rules of Fig. 1. These rules have the form

$$\frac{\Delta_1 \vdash A_1 \qquad \ldots \qquad \Delta_n \vdash A_n}{\Delta \vdash A}$$

(with possibly $n = 0$). We say that $\Delta \vdash A$ is the **conclusion**, while the sequents $\Delta_1 \vdash A_1, \ldots, \Delta_n \vdash A_n$ are the **premises** of the rule. The intended meaning of such a rule is that the conclusion holds whenever so do all the premises.

A **derivation** in NJ_0 is a finite ordered tree whose nodes are labeled by sequents and which respects the rules of Fig. 1 in the following sense:

- given a node p with list of (immediate) children $p_1, \ldots, p_n$, there is a rule

  $$\frac{\Delta_1 \vdash A_1 \qquad \ldots \qquad \Delta_n \vdash A_n}{\Delta \vdash A}$$

  such that $\Delta \vdash A$ labels p and $\Delta_i \vdash A_i$ labels p_i for each $i = 1, \ldots, n$.

$$(\mathrm{Ax})\ \frac{}{\Delta \vdash A}\ (A \in \Delta) \qquad (\top\text{-I})\ \frac{}{\Delta \vdash \top} \qquad (\bot\text{-E})\ \frac{\Delta \vdash \bot}{\Delta \vdash A}$$

$$(\Rightarrow\text{-I})\ \frac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B} \qquad (\Rightarrow\text{-E})\ \frac{\Delta \vdash A \Rightarrow B \qquad \Delta \vdash A}{\Delta \vdash B}$$

$$(\wedge\text{-I})\ \frac{\Delta \vdash A \qquad \Delta \vdash B}{\Delta \vdash A \wedge B} \qquad (\wedge_1\text{-E})\ \frac{\Delta \vdash A \wedge B}{\Delta \vdash A} \qquad (\wedge_2\text{-E})\ \frac{\Delta \vdash A \wedge B}{\Delta \vdash B}$$

$$(\vee_1\text{-I})\ \frac{\Delta \vdash A}{\Delta \vdash A \vee B} \qquad (\vee_2\text{-I})\ \frac{\Delta \vdash B}{\Delta \vdash A \vee B}$$

$$(\vee\text{-E})\ \frac{\Delta \vdash A \vee B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Delta \vdash C}$$

Figure 1: Natural Deduction for Intuitionistic Propositional Logic (NJ_0).

In particular, the leaves of a derivation must be labeled by sequents which are instances of rules with no premise.

Example 2.4. *The following are derivations in* NJ_0*.*

(1) $A \vdash A$*, using the* (Ax)*-rule* $\overline{A \vdash A}$*.*

(2) $A \wedge B \vdash B$*, with the derivation tree*

$$\frac{\dfrac{}{A \wedge B \vdash A \wedge B}}{A \wedge B \vdash A}$$

(3) $A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C$*, with the derivation tree*

$$\frac{\dfrac{\dfrac{}{A \Rightarrow B, B \Rightarrow C, A \vdash B \Rightarrow C} \qquad \dfrac{\dfrac{}{A \Rightarrow B, B \Rightarrow C, A \vdash A \Rightarrow B} \qquad \dfrac{}{A \Rightarrow B, B \Rightarrow C, A \vdash A}}{A \Rightarrow B, B \Rightarrow C, A \vdash B}}{A \Rightarrow B, B \Rightarrow C, A \vdash C}}{A \Rightarrow B, B \Rightarrow C \vdash A \Rightarrow C}$$

Exercise 2.5. *Show that the following are derivable in* NJ_0*:*

(1) $A \wedge (B \vee C) \vdash (A \wedge B) \vee (A \wedge C)$

(2) $(A \wedge B) \vee (A \wedge C) \vdash A \wedge (B \vee C)$

The following basic properties are proved by induction on derivations.

Lemma 2.6 (Structural Rules)**.** *The following properties hold in* NJ_0*:*

(weakening) *if* $\Delta \vdash A$ *then* $\Delta, B \vdash A$;

(contraction) *if* $\Delta, B, B \vdash A$ *then* $\Delta, B \vdash A$;

(exchange) *if* $\Delta, B, \Delta', C, \Delta'' \vdash A$ *then* $\Delta, C, \Delta', B, \Delta'' \vdash A$.

Lemma 2.7 (Substitution)**.** *In* NJ_0*, if* $\Delta \vdash A$ *then* $\Delta[B/\mathsf{p}] \vdash A[B/\mathsf{p}]$.

Remark 2.8. *In the following we shall **informally** speak of "intuitionistic propositional" **logic** for the **deduction system*** NJ_0*. We insist on the fact that there are different possible deduction systems for intuitionistic propositional logic (e.g. sequent calculus, Hilbert systems etc.), while natural deduction systems exist for other logics (classical, linear etc.). We refer to [SU06, §2, §5 & §6]. See also [TvD88a, Chap. 2] as well as [Bus98a, GLT89].*

The characteristic property of intuitionistic (propositional) logic is the following.

Theorem 2.9. *In intuitionistic propositional logic (*NJ_0*),*

(1) $\vdash \bot$ *is not derivable;*

(2) if $\vdash A \vee B$ *then either* $\vdash A$ *or* $\vdash B$*;*

(3) $\vdash \mathsf{p} \vee \neg\mathsf{p}$ *is **not** derivable (where* p *is an atomic proposition).*

Theorem 2.9.(2) is often called the **Disjunction Property**. We admit it until §4.4.5 (actually §5.3). On the other hand, item (3) (assuming item (2)) as well as item (1) are fairly easy (see §2.1.1 below).

2.1.1. Semantics with Truth Values

Intuitionistic propositional logic is **sound** w.r.t. the classical semantics.

Definition 2.10 (Classical Semantics)**.**

*(1) A **valuation** v is a set of atomic propositions.*

*(2) We define the relation $v \models A$ (read "v is a **model** of A" or "A is **valid under** v") by induction on A as follows:*

$$\begin{array}{lll} v \models \mathsf{p} & \textit{iff} & \mathsf{p} \in v \\ v \models A \Rightarrow B & \textit{iff} & (v \models A \textit{ implies } v \models B) \\ v \models A \wedge B & \textit{iff} & (v \models A \textit{ and } v \models B) \\ v \models A \vee B & \textit{iff} & (v \models A \textit{ or } v \models B) \\ v \models \top & & \\ v \not\models \bot & & \end{array}$$

*(3) A formula A is **valid** when A is valid under all valuations.*

*(4) A sequent $A_1, \ldots, A_n \vdash A$ is **valid** if the formula $(A_1 \wedge \cdots \wedge A_n) \Rightarrow A$ is valid.*

Remark 2.11 (Soundness w.r.t. Classical Semantics)**.** *It is easy to see that if* NJ_0 *proves* $\Delta \vdash A$ *then* $\Delta \vdash A$ *is valid.*

Beware that the converse of Rem. 2.11 fails: the formula $\mathsf{p} \vee \neg\mathsf{p}$ is valid but is not provable in NJ_0 (Thm. 2.9.(3)). See §2.3 for the case of classical logic.

Intuitionistic (propositional) logic is complete only w.r.t. weaker notions of validity, relying on stronger semantics (see [SU06, §2.4 & 2.5], also [vD04, TvD88a, TvD88b]). Such semantics in particular give the disjunction property (Thm. 2.9.(2)) without relying on the Curry-Howard correspondence (in contrast with our approach in §4.4), see [SU06, Prop. 2.5.9] (see also [vD04, Thm. 5.4.2]).

Remark 2.12. *Soundness w.r.t. classical semantics (Rem. 2.11) trivially gives item (1) of Thm. 2.9. It also easily gives item (3) assuming item (2).*

Proof. Assume that $\vdash \mathsf{p}\vee\neg\mathsf{p}$ is derivable in NJ_0. Then Thm. 2.9.(2) implies that either $\vdash \mathsf{p}$ or $\vdash \neg\mathsf{p}$ is derivable in NJ_0. But this is a contradiction since neither p nor $\neg\mathsf{p}$ is valid in the classical semantics. □

A result which is typically proved using (specific) truth values for intuitionistic propositional logic is the decidability of provability in NJ_0. See [SU06, Thm. 2.4.12] (also [vD04, Ex. 15]).

Theorem 2.13. *Provability in* NJ_0 *is decidable.*

2.1.2. Derivable and Admissible Rules

We say that a rule

$$\frac{\Delta_1 \vdash A_1 \qquad \dots \qquad \Delta_n \vdash A_n}{\Delta \vdash A}$$

is **derivable** in a given system (say NJ_0) if there exists a (partial) derivation of $\Delta \vdash A$ in which leaves among $\Delta_1 \vdash A_1, \dots, \Delta_n \vdash A_n$ are allowed.

A rule as above is **admissible** if its conclusion is derivable whenever all its premises are derivable.

Example 2.14 (Cut Rule)**.** *The rule*

$$(\textsc{Cut})\ \frac{\Delta \vdash A \qquad \Delta, A \vdash B}{\Delta \vdash B}$$

is derivable as

$$\frac{\dfrac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B} \qquad \Delta \vdash A}{\Delta \vdash B}$$

Remark 2.15. *In the following, we use instances of Lem. 2.6 and of Ex. 2.14 without explicit reference.*

2.2. Intuitionistic Negation

Intuitionistic negation is an interesting connective, in particular because in general NJ_0 does not prove $\neg\neg A \vdash A$ (where $\neg A$ is defined in Notation 2.1).

Remark 2.16. *In the following (and in accordance with e.g. [TvD88a, Chap. 2, §3]), we record which properties use the* (ExFalso) *rule (denoted* ($\bot$-E) *in Fig. 1). Note that removing* (ExFalso) *from* NJ_0 *results in having* ***no*** *rule for* $\bot$.

Intuitionistic logic without (ExFalso) *is called* ***minimal logic*** *([SU06, 2.2.2], see also [TvD88a, Chap. 2, Def. 3.2]). The implicational fragment of minimal logic is discussed in §2.4.*

The following gathers simple basic but important properties of intuitionistic negation.

Lemma 2.17. *The following are derivable in* NJ_0 *(without* (ExFalso)*):*

(1) $B \Rightarrow \neg A \vdash A \Rightarrow \neg B$

(2) $A \vdash \neg\neg A$

(3) $A \Rightarrow B \vdash \neg B \Rightarrow \neg A$

(4) $A \Rightarrow B \vdash \neg\neg A \Rightarrow \neg\neg B$

(5) $\neg\neg\neg A \vdash \neg A$

Proof. We use the (Cut) rule (Ex. 2.14) without explicit reference.

(1) Since NJ_0 (without (ExFalso)) proves $B \Rightarrow \neg A, A, B \vdash \bot$.

(2) By (1) and the fact that NJ_0 proves $\vdash \neg A \Rightarrow \neg A$.

(3) By (2) we get $A \Rightarrow B \vdash A \Rightarrow \neg\neg B$. Then conclude with (1).

(4) By (3) twice.

(5) By (2) twice we get $A \Rightarrow \neg\neg\neg\neg A$. Then conclude with (1). □

Remark 2.18. *In the following, we shall often refer to Lem. 2.17 for its combination with Rem. 2.15 (§2.1.2), for instance resulting in the admissible rule*

$$\frac{\Delta, A \vdash B}{\Delta, \neg\neg A \vdash \neg\neg B}$$

obtained as

$$\begin{prooftree}
\AxiomC{$A \Rightarrow B \vdash \neg\neg A \Rightarrow \neg\neg B$}
\UnaryInfC{$\vdash (A \Rightarrow B) \Rightarrow (\neg\neg A \Rightarrow \neg\neg B)$}
\doubleLine\dashedLine
\UnaryInfC{$\Delta \vdash (A \Rightarrow B) \Rightarrow (\neg\neg A \Rightarrow \neg\neg B)$}
\AxiomC{$\Delta, A \vdash B$}
\UnaryInfC{$\Delta \vdash A \Rightarrow B$}
\BinaryInfC{$\Delta \vdash \neg\neg A \Rightarrow \neg\neg B$}
\doubleLine\dashedLine
\UnaryInfC{$\Delta, \neg\neg A \vdash \neg\neg A \Rightarrow \neg\neg B$}
\AxiomC{}
\UnaryInfC{$\Delta, \neg\neg A \vdash \neg\neg A$}
\BinaryInfC{$\Delta, \neg\neg A \vdash \neg\neg B$}
\end{prooftree}$$

2.2.1. Basic Classical Laws: Excluded Middle and Elimination of Double Negation

We now discuss the extension of NJ_0 with some classical principles. This §2.2.1 focuses on the **excluded middle** and on the **elimination of double negation**. **Reductio ad absurdum** (see §1) as well as the more general **Peirce's law** are discussed in §2.2.2. In both cases we refer to [SU06, §6.1].

Definition 2.19 (Basic Classical Laws)**.**

- *The* ***law of excluded middle*** *is the rule*

$$(\mathrm{EM})\ \dfrac{}{\Delta \vdash A \vee \neg A}$$

- *The* ***double-negation elimination law*** *is the rule*

$$(\mathrm{DNE})\ \dfrac{}{\Delta \vdash \neg\neg A \Rightarrow A}$$

Lemma 2.20. NJ_0 *(without* (ExFalso)*) proves* $\vdash \neg\neg(A \vee \neg A)$*.*

Proof. We give the proof tree:

$$\dfrac{\dfrac{\dfrac{}{\neg(A \vee \neg A) \vdash \neg(A \vee \neg A)} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\neg(A \vee \neg A) \vdash \neg(A \vee \neg A)} \qquad \dfrac{\dfrac{}{\neg(A \vee \neg A), A \vdash A}}{\neg(A \vee \neg A), A \vdash A \vee \neg A}}{\neg(A \vee \neg A), A \vdash \bot}}{\neg(A \vee \neg A) \vdash \neg A}}{\neg(A \vee \neg A) \vdash A \vee \neg A}}{\neg(A \vee \neg A) \vdash \bot}}{\vdash \neg\neg(A \vee \neg A)}$$

□

Corollary 2.21. NJ_0 *(without* (ExFalso) *but) augmented with* (DNE) *proves all instances of* (EM)*.*

Lemma 2.22. NJ_0 *(**with*** (ExFalso) *and) augmented with* (EM) *proves all instances of* (DNE)*.*

Proof. We give the proof tree:

$$\dfrac{\dfrac{\vdash A \vee \neg A \qquad \dfrac{}{\neg\neg A, A \vdash A} \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \neg\neg A} \qquad \dfrac{}{\neg\neg A, \neg A \vdash \neg A}}{\neg\neg A, \neg A \vdash \bot}}{\neg\neg A, \neg A \vdash A}}{\neg\neg A \vdash A}}{\vdash \neg\neg A \Rightarrow A}$$

□

Corollary 2.23. *The following systems prove the same sequents:*

(i) NJ_0 ***(with** (*EXFALSO*) and) augmented with* (DNE)*;*

(ii) NJ_0 ***(with** (*EXFALSO*) and) augmented with* (EM)*.*

Remark 2.24 (Non Interdefinability of Connectives)**.** *Assuming that there is at least one atomic proposition, in intuitionistic logic connectives are **not** interdefinable as in classical logic. In each case below, there are formulae A, B such that the indicated sequents are **not** derivable in* NJ_0*:*

(i) $(A \Rightarrow B) \vdash \neg A \vee B$

(ii) $\neg(\neg A \wedge \neg B) \vdash A \vee B$

(iii) $\neg(\neg A \vee \neg B) \vdash A \wedge B$

PROOF. In each case, we find instances of A and B which contradict Thm. 2.9.(3).

(i) We take $A = B = \mathsf{p}$, an atomic proposition. Then $\mathsf{p} \Rightarrow \mathsf{p}$ is provable in NJ_0 but not $\neg\mathsf{p} \vee \mathsf{p}$.

(ii) Assume that NJ_0 proves $\neg(\neg A \wedge \neg A) \vdash A \vee A$ for all A. Then NJ_0 proves all instances of (DNE) (since NJ_0 proves $A \Leftrightarrow (A \wedge A)$ and $A \Leftrightarrow (A \vee A)$), which by Cor. 2.23 implies that NJ_0 proves all instances of (EM), a contradiction.

(iii) Similar. $\square$

2.2.2. Reductio ad Absurdum and The Law of Peirce

We now complete the investigation of classical principles initiated in §2.2.1. We still refer to [SU06, §6.1].

Definition 2.25.

- *The **law of Peirce** is the rule*

$$(\textsc{Peirce})\ \frac{}{\Delta \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

- *The principle of **reductio ad absurdum** is the rule*

$$(\mathrm{RAA})\ \frac{}{\Delta \vdash (\neg A \Rightarrow A) \Rightarrow A}$$

The principle (RAA) was discussed informally in §1. Its generalization to Peirce's law is crucial w.r.t. **Minimal Logic** (see §2.4 below).

Lemma 2.26. NJ_0 *(without* (*EXFALSO*) *but) augmented with either* (EM) *or* (DNE) *proves all instances of* (RAA)*.*

PROOF. In the case of $\mathsf{NJ}_0 +$ (EM), the proof tree is

$$\dfrac{\vdash A \lor \neg A \qquad \dfrac{}{\neg A \Rightarrow A, A \vdash A} \qquad \dfrac{\dfrac{}{\neg A \Rightarrow A, \neg A \vdash \neg A \Rightarrow A} \qquad \dfrac{}{\neg A \Rightarrow A, \neg A \vdash \neg A}}{\neg A \Rightarrow A, \neg A \vdash A}}{\dfrac{\neg A \Rightarrow A \vdash A}{\vdash (\neg A \Rightarrow A) \Rightarrow A}}$$

Together with Cor. 2.21, this also gives the result for $\mathsf{NJ}_0 + (\text{DNE})$. □

Lemma 2.27. NJ_0 *(**with** (*Ex<span style="font-variant:small-caps">Falso</span>*) and) augmented with* (RAA) *proves all instances of* (Peirce)*,* (DNE) *and* (EM)*.*

Proof. In the case of (Peirce), we give the proof tree:

$$\dfrac{\vdash (\neg A \Rightarrow A) \Rightarrow A \qquad \dfrac{\dfrac{\dfrac{}{(A \Rightarrow B) \Rightarrow A, \neg A \vdash (A \Rightarrow B) \Rightarrow A} \qquad \dfrac{\dfrac{\dfrac{}{(A \Rightarrow B) \Rightarrow A, \neg A, A \vdash \bot}}{(A \Rightarrow B) \Rightarrow A, \neg A, A \vdash B}}{(A \Rightarrow B) \Rightarrow A, \neg A \vdash A \Rightarrow B}}{(A \Rightarrow B) \Rightarrow A, \neg A \vdash A}}{(A \Rightarrow B) \Rightarrow A \vdash \neg A \Rightarrow A}}{\dfrac{(A \Rightarrow B) \Rightarrow A \vdash A}{\vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}}$$

For the remaining cases, we only consider (DNE) (and conclude with Cor. 2.23):

$$\dfrac{\vdash (\neg A \Rightarrow A) \Rightarrow A \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg A, \neg A \vdash \bot}}{\neg\neg A, \neg A \vdash A}}{\neg\neg A \vdash \neg A \Rightarrow A}}{\dfrac{\neg\neg A \vdash A}{\vdash \neg\neg A \Rightarrow A}}$$

□

Corollary 2.28. *The following systems prove the same sequents:*

(i) NJ_0 *(**with** (*ExFalso*) and) augmented with* (Peirce)*;*

(ii) NJ_0 *(**with** (*ExFalso*) and) augmented with* (RAA)*;*

(iii) NJ_0 *(**with** (*ExFalso*) and) augmented with* (DNE)*;*

(iv) NJ_0 *(**with** (*ExFalso*) and) augmented with* (EM)*.*

It follows from Thm. 2.9 that the equivalent (w.r.t. provability) systems of Cor. 2.28 are proper extensions of NJ_0.

Remark 2.29. *Note that some instances of seemingly classical principles are actually intuitionistic. We have already seen in Lem. 2.17.(5) that* NJ_0 *admits elimination of double negation over negated formulae. The same is true for* (RAA)*:*

- NJ_0 *(without* (ExFalso)*) proves* $\vdash (\neg\neg A \Rightarrow \neg A) \Rightarrow \neg A$.

Proof. Exercise! □

2.3. Classical Propositional Logic

The system NK_0 of natural deduction for **classical** propositional logic is NJ_0 augmented with the rule (EM). The system NK_0 is complete w.r.t. the classical semantics (Def. 2.10, §2.1.1).

Theorem 2.30 (Completeness of NK_0)**.** *NK_0 proves all valid sequents.*

Theorem 2.30 can be obtained in different ways. We give a proof in Appendix A (Thm A.9, §A.3), which uses a Gentzen-style **sequent calculus** (see [SU06, §7], [GLT89, §5], [Bus98a, §1.2] or [Mel09, §1]).

Remark 2.31 (Compactness and Related Results)**.** *The reader is invited to look at related results (including* ***compactness****), e.g. [SU06, Thm. 6.1.10] or [Bus98a, Thm. 1.1.3 & Thm. 1.1.5], also [vD04, Thm. 1.5.13]. See §A.4 for a discussion.*

2.4. Minimal Implicational Logic

We make a detour via a very simple logic called **minimal implicational logic** (see Rem. 2.16, §2.2). Its formulae are given by the grammar

$$A, B \quad ::= \quad \mathsf{p} \quad | \quad A \Rightarrow B$$

where p ranges over some (unspecified) set of atomic propositions.

Remark 2.32 (Terminology)**.** *Hence minimal implicational logic is the implicational fragment of minimal logic as defined in Rem. 2.16. In the remaining of this §2.4 we write "minimal logic" for "minimal implicational logic".*

Deduction for **intuitionistic** minimal (implicational) logic is given by the rules of NJ_0 restricted to the connectives of minimal logic, namely:

$$(\text{Ax})\ \frac{A \in \Delta}{\Delta \vdash A} \qquad\qquad (\Rightarrow\text{-I})\ \frac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B} \qquad\qquad (\Rightarrow\text{-E})\ \frac{\Delta \vdash A \Rightarrow B \qquad \Delta \vdash A}{\Delta \vdash B}$$

Note in particular that (ExFalso) is not available, for the reason that there is no $\bot$ in minimal logic! Also, among the classical principles mentioned in Cor. 2.28, only Peirce's law is available for minimal logic. We thus define deduction for **classical** minimal logic to consist of deduction for intuitionistic minimal logic augmented with the rule (Peirce) of Def. 2.25 (§2.2.2):

$$(\textsc{Peirce})\ \frac{}{\Delta \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

As we shall see in §4.3, minimal logic is the core of the Curry-Howard correspondence. It is also of central importance for Second-Order Logic (see §**??**).

For the moment we only establish two basic facts about minimal logic.

Proposition 2.33. *Assume that there is exactly one atomic proposition. Then classical and intuitionistic minimal logics prove the same sequents.*

Proof. Exercise! □

Proposition 2.34. *Assume that* p *and* q *are* ***distinct*** *atomic propositions. Then intuitionistic minimal logic does* ***not*** *prove* $\vdash ((\mathsf{p} \Rightarrow \mathsf{q}) \Rightarrow \mathsf{p}) \Rightarrow \mathsf{p}$.

Proof. Exercise! □

Remark 2.35. *Note that the proof of Prop. 2.34 apparently uses the classical* (RAA)*. But it is actually intuitionistic since we are proving a negated statement (see Rem. 2.29).*

Remark 2.36. *One can show that* NJ_0 *is* ***conservative*** *over intuitionistic minimal implicational logic ([SU06, Thm. 2.6.2]), so that the decidability of provability in* NJ_0 *(Thm. 2.13, §2.1.1) implies the decidability of provability in intuitionistic minimal implicational logic.*

2.5. A Negative Translation

There are **many** translations of classical logic to intuitionistic logic in the literature. We study here one such translation, which is a variant of the Gödel-Gentzen translation, see [TvD88a, Chap. 2, §3.4–8]. See also [vD04, Def. 5.2.7], [SU06, §6.4] or [Koh08, §10.1].

Definition 2.37 (Negative Translation)**.** *The formula* $A^\neg$ *is defined by induction on* A *as follows:*

$$\begin{array}{rcl} \mathsf{p}^\neg & := & \neg\neg\mathsf{p} \\ \top^\neg & := & \top \\ \bot^\neg & := & \bot \\ (A \Rightarrow B)^\neg & := & A^\neg \Rightarrow B^\neg \\ (A \wedge B)^\neg & := & A^\neg \wedge B^\neg \\ (A \vee B)^\neg & := & \neg\neg(A^\neg \vee B^\neg) \end{array}$$

Lemma 2.38. *For each formula* A*,* NJ_0 *proves* $\neg\neg A^\neg \vdash A^\neg$*.*

Proof. The proof is by induction on A. The case of $\top$ is trivial since $\Delta \vdash \top$ is always derivable with the rule ($\top$-I). The cases of p and $A \vee B$ follow from Lem. 2.17.(5). The case of $\bot$ is a kind of trick: we have

$$\begin{array}{rcl} \neg\neg\bot & = & \neg(\bot \Rightarrow \bot) \\ & = & (\bot \Rightarrow \bot) \Rightarrow \bot \end{array}$$

Hence $\neg\neg\bot \vdash \bot$ since $\bot \Rightarrow \bot$ is provable in NJ_0.

The remaining cases ($A \wedge B$ and $A \Rightarrow B$) use the induction hypothesis:

Case of $A \wedge B$**.** We have to derive $\neg\neg(A^\neg \wedge B^\neg) \vdash A^\neg \wedge B^\neg$. We have $A^\neg \wedge B^\neg \vdash A^\neg$ in NJ_0 and thus $\neg\neg(A^\neg \wedge B^\neg) \vdash \neg\neg A^\neg$ by Lem. 2.17.(4). The induction hypothesis then gives $\neg\neg(A^\neg \wedge B^\neg) \vdash A^\neg$. We similarly obtain $\neg\neg(A^\neg \wedge B^\neg) \vdash B^\neg$ and conclude with ($\wedge$-I).

Case of $A \Rightarrow B$. We have to derive $\neg\neg(A^\neg \Rightarrow B^\neg) \vdash A^\neg \Rightarrow B^\neg$. By induction hypothesis, we are done if we derive $\neg\neg(A^\neg \Rightarrow B^\neg), A^\neg \vdash \neg\neg B^\neg$. But it is easy to see that $A^\neg, \neg B^\neg, A^\neg \Rightarrow B^\neg \vdash \bot$, so that $A^\neg, \neg B^\neg \vdash \neg(A^\neg \Rightarrow B^\neg)$ and thus $\neg\neg(A^\neg \Rightarrow B^\neg), A^\neg, \neg B^\neg \vdash \bot$. □

Theorem 2.39. *If $A_1, \dots, A_n \vdash A$ is derivable in NK_0, then $A_1^\neg, \dots, A_n^\neg \vdash A^\neg$ is derivable in NJ_0.*

Proof. By induction on derivations. We reason by case on the last applied rule. But for each rule

$$\frac{\Delta_1 \vdash A_1 \quad \dots \quad \Delta_n \vdash A_n}{\Delta \vdash A}$$

excepted those concerning disjunction (($\vee_1$-I), ($\vee_2$-I), ($\vee$-E) and (EM)) we have in NJ_0

$$\frac{\Delta_1^\neg \vdash A_1^\neg \quad \dots \quad \Delta_n^\neg \vdash A_n^\neg}{\Delta^\neg \vdash A^\neg}$$

and the result follows from the induction hypothesis. We thus only discuss the remaining rules:

Cases of ($\vee_1$-I) and ($\vee_2$-I):

$$(\vee_1\text{-I})\ \frac{\Delta \vdash A}{\Delta \vdash A \vee B} \qquad\qquad (\vee_2\text{-I})\ \frac{\Delta \vdash B}{\Delta \vdash A \vee B}$$

We only discuss ($\vee_1$-I). The induction hypothesis gives $\Delta^\neg \vdash A^\neg$ and thus $\Delta^\neg \vdash A^\neg \vee B^\neg$. We then conclude with Lem. 2.17.(2).

Case of ($\vee$-E):

$$(\vee\text{-E})\ \frac{\Delta \vdash A \vee B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Delta \vdash C}$$

The induction hypothesis (applied three times) gives $\Delta^\neg \vdash \neg\neg(A^\neg \vee B^\neg)$ and $\Delta^\neg, A^\neg \vdash C^\neg$ as well as $\Delta^\neg, B^\neg \vdash C^\neg$.

Note that NJ_0 proves

$$A^\neg \vee B^\neg,\ A^\neg \Rightarrow C^\neg,\ B^\neg \Rightarrow C^\neg \quad\vdash\quad C^\neg$$

and thus (using Lem. 2.17.(4), see Rem. 2.18)

$$\neg\neg(A^\neg \vee B^\neg),\ A^\neg \Rightarrow C^\neg,\ B^\neg \Rightarrow C^\neg \quad\vdash\quad \neg\neg C^\neg$$

We then obtain $\Delta^\neg \vdash \neg\neg C^\neg$ and conclude with Lem. 2.38.

Case of (EM):

$$(\text{EM})\ \frac{}{\Delta \vdash A \vee \neg A}$$

By Lem. 2.20. □

Corollary 2.40. *If* NK_0 *proves* $\vdash A$ *then* NJ_0 *proves* $\vdash A^{\neg}$.

Remark 2.41 (On Minimal Logic)**.** *Corollary 2.40 (i.e. Thm. 2.39) can be strengthened to* ***minimal*** *logic (Rem. 2.16), in the sense that if* $\vdash A$ *in* NK_0 *then* $\vdash A^{\neg}$ *can be derived in* NJ_0 *without* (ExFalso)*. To this end, it suffices to observe that the statement of Lem. 2.38 actually holds in minimal logic and that* $\bot \vdash A^{\neg}$ *is provable in minimal logic for each formula* A*. We refer to [TvD88a, Chap. 2, §3] for details.*

Remark 2.42. *As we shall see in §6.2.3 and §7.3.4, the translation* $(-)^{\neg}$ *generalizes to stronger systems, and can be extended so as to provide extraction results (in the sense outlined in §1).*

Further, under the ***Curry-Howard correspondence****, variants of* $(-)^{\neg}$ *extend to* ***continuation-passing style*** *(CPS) translations for (extensions of) the* λ*-calculus. For more on this, see [SU06, §6] (and also [Kri09] and [Sel01, §5–8]).*

Remark 2.43 (On the Gödel-Gentzen and Kolmogorov Translations)**.** *The translation* $(-)^{\neg}$ *devised in Def. 2.37 makes transparent the case of* (EM) *(and of* (DNE)*) via the clause*

$$(A \vee B)^{\neg} \quad := \quad \neg\neg(A^{\neg} \vee B^{\neg})$$

(and Lem. 2.38), in particular in view of Lem. 2.20 (§2.2.1). In contrast, the original Gödel-Gentzen translation (see e.g. [TvD88a, Chap. 2, Def. 3.4] or [vD04, Def. 5.2.7]) assumes

$$(A \vee B)^{\neg} \quad := \quad \neg(\neg A^{\neg} \wedge \neg B^{\neg})$$

This however does not substantially change the proof of Lem. 2.38 and Thm. 2.39.

On the other hand, in view of Rem. 2.42 it is desirable to consider variants of $(-)^{\neg}$ *which avoid Lem. 2.38. This is the case for instance of the following* ***Kolmogorov translation*** $(-)^{\neg\neg}$ *(see [TvD88a, Chap. 2, 3.7], and also [SU06, §6.4]):*

$$\begin{array}{rcl}
\mathsf{p}^{\neg\neg} & := & \neg\neg\mathsf{p} \\
\top^{\neg\neg} & := & \neg\neg\top \\
\bot^{\neg\neg} & := & \neg\neg\bot \\
(A \Rightarrow B)^{\neg\neg} & := & \neg\neg(A^{\neg\neg} \Rightarrow B^{\neg\neg}) \\
(A \wedge B)^{\neg\neg} & := & \neg\neg(A^{\neg\neg} \wedge B^{\neg\neg}) \\
(A \vee B)^{\neg\neg} & := & \neg\neg(A^{\neg\neg} \vee B^{\neg\neg})
\end{array}$$

(we differ from [TvD88a, Chap. 2, 3.7] in the case of $\top$*). The crucial difference (with both variants of* $(-)^{\neg}$*) is that* $(-)^{\neg\neg}$ *relies on the uniform proof of* $\neg\neg\neg A \vdash \neg A$ *(Lem. 2.17.(5), §2.2) instead of the inductive and (thus) non-uniform derivations of Lem. 2.38.*

2.6. Glivenko's Theorem

We now discuss **Glivenko's Theorem**, a well-known result which establishes that

$$\mathsf{NK}_0 \text{ proves } \vdash A \qquad \text{if and only if} \qquad \mathsf{NJ}_0 \text{ proves } \vdash \neg\neg A$$

(see [SU06, Thm. 2.4.10] (proved by semantic methods), or *e.g.* [vD04, Thm. 5.2.10]). At first sight, this result may seem to make the negative translation $(-)^\neg$ useless. Beware that the situation is more subtle, chiefly because Glivenko's Theorem does **not** extend to (full) first-order logic, but also for reasons discussed in Rem. 2.50 below.

Lemma 2.44. *NJ_0 proves $\neg\neg A, \neg\neg B \vdash \neg\neg(A \wedge B)$.*

Proof. We give a proof tree:

$$\dfrac{\dfrac{\dfrac{}{\neg\neg A \vdash \neg\neg A} \qquad \dfrac{\dfrac{\dfrac{}{\neg\neg B \vdash \neg\neg B} \qquad \dfrac{\dfrac{\dfrac{}{\neg(A\wedge B) \vdash \neg(A \wedge B)} \qquad A, B \vdash A \wedge B}{\neg\neg A, \neg\neg B, \neg(A\wedge B), A, B \vdash \bot}}{\neg\neg A, \neg\neg B, \neg(A \wedge B), A \vdash \neg B}}{\neg\neg A, \neg\neg B, \neg(A\wedge B), A \vdash \bot}}{\neg\neg A, \neg\neg B, \neg(A \wedge B) \vdash \neg A}}{\neg\neg A, \neg\neg B, \neg(A \wedge B) \vdash \bot}}{\neg\neg A, \neg\neg B \vdash \neg\neg(A \wedge B)}$$

$\square$

Lemma 2.45. *NJ_0 proves $(\neg\neg A \Rightarrow \neg\neg B) \vdash \neg\neg(A \Rightarrow B)$.*

Proof. First, note that (using (ExFalso)) we have $\neg A \vdash A \Rightarrow B$. It follows that $\neg\neg A \Rightarrow \neg\neg B, \neg(A \Rightarrow B) \vdash \neg\neg B$, with proof tree *e.g.*:

$$\dfrac{\dfrac{}{\neg\neg A \Rightarrow \neg\neg B \vdash \neg\neg A \Rightarrow \neg\neg B} \qquad \dfrac{\dfrac{\dfrac{}{\neg(A \Rightarrow B) \vdash \neg(A \Rightarrow B)} \qquad \neg A \vdash A \Rightarrow B}{\neg(A \Rightarrow B), \neg A \vdash \bot}}{\neg(A \Rightarrow B) \vdash \neg\neg A}}{\neg\neg A \Rightarrow \neg\neg B, \neg(A \Rightarrow B) \vdash \neg\neg B}$$

Since on the other hand $B \vdash A \Rightarrow B$, we can conclude with the proof tree:

$$\dfrac{\dfrac{\neg\neg A \Rightarrow \neg\neg B, \neg(A \Rightarrow B) \vdash \neg\neg B \qquad \dfrac{\dfrac{\dfrac{}{\neg(A \Rightarrow B) \vdash \neg(A \Rightarrow B)} \qquad B \vdash A \Rightarrow B}{\neg(A \Rightarrow B), B \vdash \bot}}{\neg(A \Rightarrow B) \vdash \neg B}}{\neg\neg A \Rightarrow \neg\neg B, \neg(A \Rightarrow B) \vdash \bot}}{\neg\neg A \Rightarrow \neg\neg B \vdash \neg\neg(A \Rightarrow B)}$$

$\square$

Lemma 2.46. *NJ_0 proves the following:*

(1) $\vdash (\neg\neg A \wedge \neg\neg B) \Leftrightarrow \neg\neg(A \wedge B)$

(2) $\vdash (\neg\neg A \Rightarrow \neg\neg B) \Leftrightarrow \neg\neg(A \Rightarrow B)$

(3) $\vdash \neg\neg(A \vee B) \Rightarrow (\neg\neg A \Rightarrow \neg\neg C) \Rightarrow (\neg\neg B \Rightarrow \neg\neg C) \Rightarrow \neg\neg C$

Proof.

(1) We have $\vdash (\neg\neg A \wedge \neg\neg B) \Rightarrow \neg\neg(A \wedge B)$ by Lem. 2.44. The other direction follows from Lem. 2.17.(4) applied twice.

(2) First, by Lem. 2.45 we have $\vdash (\neg\neg A \Rightarrow \neg\neg B) \Rightarrow \neg\neg(A \Rightarrow B)$.

We now show $\vdash \neg\neg(A \Rightarrow B) \Rightarrow \neg\neg A \Rightarrow \neg\neg B$. We show $\vdash (\neg\neg(A \Rightarrow B) \wedge \neg\neg A) \Rightarrow \neg\neg B$. We trivially have $\vdash ((A \Rightarrow B) \wedge A) \Rightarrow B$, so that Lem. 2.17.(4) gives $\vdash \neg\neg((A \Rightarrow B) \wedge A) \Rightarrow \neg\neg B$, and we conclude by item (1).

(3) First, note that $\vdash (A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C$, from which Lem. 2.17.(2) gives $\vdash \neg\neg((A \vee B) \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C)$. We then conclude by iterated applications of item (2). □

Proposition 2.47. *If $A_1, \ldots, A_n \vdash A$ in NK_0 then $\neg\neg A_1, \ldots, \neg\neg A_n \vdash \neg\neg A$ in NJ_0.*

Proof. We reason by induction on derivations and by cases on the last applied rule.

Case of

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

We have $\neg\neg\Delta \vdash \neg\neg(A \vee \neg A)$ by Lem. 2.20.

Case of

$$\frac{A \in \Delta}{\Delta \vdash A}$$

Trivial.

Case of

$$\frac{}{\Delta \vdash \top}$$

We have $\neg\neg\Delta \vdash \top$ and we conclude with Lem. 2.17.(2).

Case of

$$\frac{\Delta \vdash \bot}{\Delta \vdash A}$$

The induction hypothesis gives $\neg\neg\Delta \vdash \neg\neg\bot$. Since $\bot \vdash A$ we can conclude with Lem. 2.17.(4).

Cases of

$$\frac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B} \qquad \frac{\Delta \vdash A \Rightarrow B \qquad \Delta \vdash A}{\Delta \vdash B}$$

$$\frac{\Delta \vdash A \qquad \Delta \vdash B}{\Delta \vdash A \wedge B} \qquad \frac{\Delta \vdash A \wedge B}{\Delta \vdash A} \qquad \frac{\Delta \vdash A \wedge B}{\Delta \vdash B}$$

By induction hypothesis and Lem. 2.46 (1) and (2).

Cases of

$$\frac{\Delta \vdash A}{\Delta \vdash A \vee B} \qquad \frac{\Delta \vdash B}{\Delta \vdash A \vee B}$$

By induction hypothesis and Lem. 2.17.(4).

Case of

$$\frac{\Delta \vdash A \vee B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Delta \vdash C}$$

By induction hypothesis and Lem. 2.46.(3). □

Glivenko's Theorem is an immediate consequence of Prop. 2.47.

Theorem 2.48 (Glivenko)**.** NK_0 *proves* $\vdash A$ *if and only if* NJ_0 *proves* $\vdash \neg\neg A$*.*

An interesting consequence of Glivenko's Theorem 2.48 is that NK_0 and NJ_0 negate the same formulae.

Corollary 2.49. NK_0 *proves* $A \vdash \bot$ *if and only if* NJ_0 *proves* $A \vdash \bot$*.*

Proof. If NJ_0 proves $A \vdash \bot$ then obviously NK_0 proves $A \vdash \bot$. Conversely, if NK_0 proves $A \vdash \bot$ then NK_0 proves $\vdash \neg A$. Hence NJ_0 proves $\vdash \neg\neg\neg A$ by Glivenko's Theorem 2.48, so that NJ_0 proves $\vdash \neg A$ by Lem. 2.17.(5). But it then easily follows that NJ_0 proves $A \vdash \bot$:

$$\frac{\vdash \neg A \qquad \dfrac{}{A \vdash A}}{A \vdash \bot}$$

□

Remark 2.50. *In comparison with negative translations (such as the translation* $(-)^\neg$ *discussed in §2.5 above), the main limitation of Glivenko's Theorem 2.48 is that it does* ***not*** *extend to full first-order logic (we shall discuss this point in §6.2.5).*

Besides, the transformation of proofs underlying Thm. 2.48 lacks structure to be (as such) directly exploitable under the Curry-Howard correspondence. The reason is that Lem. 2.44 actually admits two canonical proofs which cannot be equated at the level of the λ*-calculus (without loosing the consistency of its equational theory). This phenomenon has been analyzed in [Gir91], which led to a deep analysis of CPS translations (Rem. 2.42). See [Mel17, §5.7] (in French) for more.*

3. The Untyped Lambda-Calculus

3.1. Introduction

The λ**-calculus** was invented in 1936 by Alonzo Church, originally with the intent to provide a foundational logical framework based on the notion of **function**. However, the (pure) untyped λ-calculus turned out to be inconsistent as a logical system, due to the presence of **fixpoints**. On the other hand, the latter make the untyped λ-calculus a **Turing-complete** model of computation, which constitutes the (theoretical) core of functional traits of programming languages. Moreover, suitable restrictions on the formation of λ-terms (most notably based on **types**) make it possible to recover (a range of) logically consistent systems.

References. An excellent general but synthetic introduction to the (pure) untyped λ-calculus is [Bar92, §2]. The standard reference book is [Bar84]. Other important presentations are [AC98, Chap. 2] and [Kri93]. On the other hand, we shall hardly go beyond the material covered in [SU06, Chap. 1].

Idea. The λ-calculus can be thought about as a formal system to represent and manipulate functions. For instance, the function

$$f \quad : \quad x \quad \longmapsto \quad x+3$$

is represented by an expression of the form:

$$\mathtt{f} \quad := \quad \lambda x.\ x+3$$

Instead of $f(4)$, the application of $\mathtt{f}$ to the argument 4 is written

$$(\lambda x.x+3)4$$

Function evaluation is performed by a syntactic manipulation of expressions called **β-reduction**, and which can be formulated as:

- *substitution of "bound variables" by actual arguments.*

For instance, it is expected that

$$(\lambda x.x+3)4 \qquad \text{evaluates to} \qquad 4+3$$

This is formally achieved as:

$$(\lambda x.x+3)4 \qquad \text{evaluates to} \qquad (x+3)[4/x]$$

where $(x+3)[4/x]$ means "$(x+3)$ in which x is replaced by 4".

Moreover, we shall identify the two following expressions, which only differ by renaming of their "bound variables", and as such are just two different notations for the same object (see §3.2.3 & §3.2.4 for details):

$$(\lambda x.x+3) \qquad \text{and} \qquad (\lambda y.y+3)$$

An important aspect of the (pure) λ-calculus is the following:

- **functions** and **values** are manipulated at the same level, in the sense that they belong to the same class of objects.

Finally, in the λ-calculus there are also **higher-order** functions, *i.e.* functions which take functions as arguments, for instance

$$\lambda f.\lambda x.fx3$$

Informally, the λ-calculus allows for the following form of computations:

$$\begin{array}{lll} (\lambda f.\lambda x.fx3)(\lambda y.\lambda z.y+z) & \text{evaluates to} & \lambda x.(\lambda y.\lambda z.y+z)x3 \\ \lambda x.(\lambda y.\lambda z.y+z)x3 & \text{evaluates to} & \lambda x.x+3 \end{array}$$

(where *e.g.* $fx3$ stands for $(fx)3$, see Notation 3.2 in §3.2 below).

3.2. Syntax

3.2.1. The Terms of the Lambda-Calculus

We assume given a countably infinite set $\mathcal{X} = \{x, y, z, \dots\}$ of **variables**. The terms of the λ-calculus (or **λ-terms**) are given by the grammar

$$t, u \in \Lambda \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad tu \qquad\qquad (\text{where } x \in \mathcal{X})$$

In the above grammar:

- Terms of the form tu are called **applications**. Formally, there is a binary term constructor @, and tu stands for $@(t, u)$.
- Terms of the form $\lambda x.t$ are called **λ-abstractions**. In $\lambda x.t$, we say that λ is a **binder** and that λx is the **binding site** of x. The variable x is said to be **bound** in $\lambda x.t$.

3.2.2. Examples and Notational Conventions

Example 3.1. *The following usual λ-terms play an important role in this course:*

$$\begin{array}{rcl} \texttt{id} & := & \lambda x.x \\ \texttt{pair} & := & \lambda x.\lambda y.\lambda k.(kx)y \\ \texttt{T} & := & \lambda x.\lambda y.x \\ \texttt{F} & := & \lambda x.\lambda y.y \\ \delta & := & \lambda x.xx \\ \Omega & := & \delta\delta \\ \texttt{S} & := & \lambda n.\lambda x.\lambda f.f(nxf) \\ \underline{n} & := & \lambda x.\lambda f.\underbrace{f(\dots(f}_{n}\,x)\dots) \quad = \quad \lambda x.\lambda f.f^n x \qquad (n \in \mathbb{N}) \end{array}$$

The λ-terms $\underline{n}$ representing $n \in \mathbb{N}$ above are called ***Church's numerals****.*

Notation 3.2. *The usual notational convention is that application associates to the left, so that*

$$xyz \qquad \textit{stands for} \qquad (xy)z \qquad \textit{(that is } @(@(x,y),z)\textit{)}$$

With this convention, each λ-term is of the form

$$\begin{array}{ll} & \lambda x_1.\dots.\lambda x_n.yt_1\dots t_m \\ \textit{or} & \lambda x_1.\dots.\lambda x_n.(\lambda y.u)tt_1\dots t_m \end{array}$$

This is called the ***Wadsworth's notation****.*

3.2.3. Free and Bound Variables, Towards Alpha-Conversion

A variable $x \in \mathcal{X}$ is **free** in a λ-term t if it does not occur in the scope of an abstraction λx in t. This is formalized with the following notion.

Definition 3.3 (Free Variable)**.** *The set* $\mathrm{FV}(t)$ *of* ***free variables*** *of t is defined by induction as follows:*

$$\begin{array}{lcl} \mathrm{FV}(x) & := & \{x\} \\ \mathrm{FV}(tu) & := & \mathrm{FV}(t) \cup \mathrm{FV}(u) \\ \mathrm{FV}(\lambda x.t) & := & \mathrm{FV}(t) \setminus \{x\} \end{array}$$

Definition 3.4 (Closed Term)**.** *A λ-term t is* ***closed*** *if* $\mathrm{FV}(t) = \emptyset$*. We write Λ_0 for the set of closed λ-terms.*

On the other hand, the notion of bound variable is much more subtle, because λ-terms are identified up-to (consistent) renaming of their bound variables. Formally, this identification is made via an equivalence relation on Λ called **α-equivalence** (or **α-conversion**) and denoted $=_\alpha$. For instance we should have

$$\lambda y.(\lambda x.xy)(xy) =_\alpha \lambda y.(\lambda z.zy)(xy)$$

but **not**

$$\lambda y.(\lambda x.xy)(xy) =_\alpha \lambda y.(\lambda y.yy)(xy) \qquad \textbf{nor} \qquad \lambda y.(\lambda x.xy)(xy) =_\alpha \lambda x.(\lambda x.xx)(xx)$$

Intuitively, λ-terms quotiented by α-equivalence can be represented as directed graphs, in which only free variables are named, while bound variables are represented by edges from their occurrences to their binding site. For instance, the α-equivalence class of $\lambda y.(\lambda x.xy)(xy)$ could be represented as the graph depicted in Fig. 2 (left). In §3.2.4 we define an algebraic (*i.e.* term) representation of such graphs.

Remark 3.5. *A convention which is often adopted in practice, but not always so (easy nor) convenient to maintain is* ***Barendregt's convention****, namely:*

(i) free and bound variables are always distinct, and

(ii) bound variables are pairwise distinct.

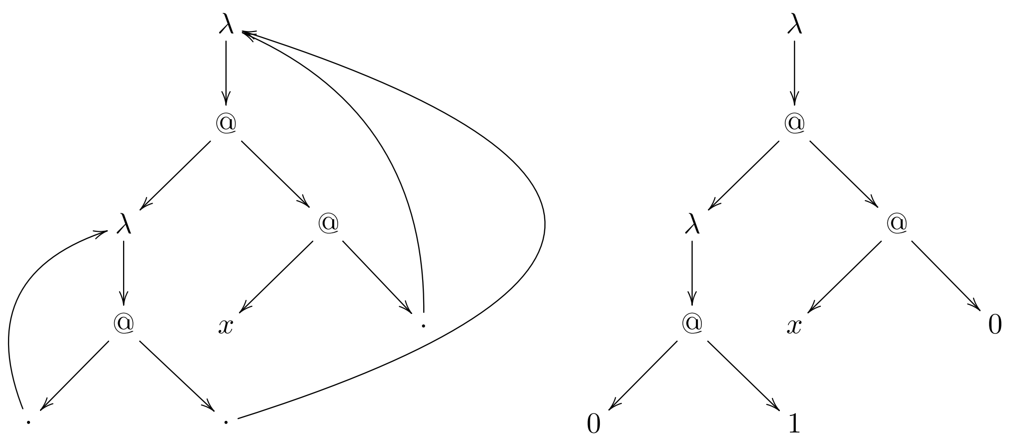

Figure 2: The Graph Representation of the α-Equivalence Class of $\lambda y.(\lambda x.xy)(xy)$.

3.2.4. Alpha-Conversion via a Locally Nameless Representation

We define α-equivalence via an algebraic representation of the graphs such as the one depicted in Fig. 2 (left). This representation uses the well-known **de Bruijn indexes** to represent bound variables, but still uses names to represent free variables. Such representations are called **locally nameless** (or sometimes **mixed**) **representations**.

The idea of de Bruijn indexes is that an **occurrence** of a bound variable is represented by a natural number, which counts the number of binders between that occurrence and its binding site. Hence, different occurrences of the **same** bound variable can be represented by **different** indexes. For instance, the locally nameless representation of $\lambda y.(\lambda x.xy)(xy)$ is depicted in Fig. 2 (right).

Formally, we consider the following **mixed de Bruijn** terms, where n ranges over natural numbers:

$$M, N \in \lambda\mathrm{DB} \quad ::= \quad x \quad | \quad n \quad | \quad \lambda M \quad | \quad MN$$

The **mixed de Bruijn representation** of a λ-term $t \in \Lambda$ is the mixed de Bruijn term $t^{\mathsf{db}} \in \lambda\mathrm{DB}$ defined by induction as follows:

$$\begin{array}{rcl} x^{\mathsf{db}} & := & x \\ (\lambda x.t)^{\mathsf{db}} & := & \lambda(t^{\mathsf{db}}[x \mapsto 0]) \\ (tu)^{\mathsf{db}} & := & t^{\mathsf{db}} u^{\mathsf{db}} \end{array}$$

where the operation $M[x \mapsto n]$ is defined by induction as

$$\begin{array}{rcll} x[x \mapsto n] & := & n & \\ y[x \mapsto n] & := & y & \text{if } y \in \mathcal{X} \text{ and } y \neq x \\ k[x \mapsto n] & := & k & \text{if } k \in \mathbb{N} \\ (MN)[x \mapsto n] & := & (M[x \mapsto n])(N[x \mapsto n]) & \\ (\lambda M)[x \mapsto n] & := & \lambda(M[x \mapsto n+1]) & \end{array}$$

For instance, we have

$$\begin{array}{rcl}(\lambda x.x)^{\mathsf{db}} & = & \lambda((x^{\mathsf{db}})[x \mapsto 0]) \\ & = & \lambda 0\end{array}$$

so that

$$((\lambda x.x)x)^{\mathsf{db}} \quad = \quad (\lambda 0)x$$

and thus

$$\begin{array}{rcl}(\lambda x.(\lambda x.x)x)^{\mathsf{db}} & = & \lambda(((\lambda 0)x)[x \mapsto 0]) \\ & = & \lambda((\lambda 0)0)\end{array}$$

Similarly, it is easy to see that in accordance with Fig. 2 (right), we have

$$(\lambda y.(\lambda x.xy)(xy))^{\mathsf{db}} \quad = \quad \lambda((\lambda(0\,1))(x0)) \quad = \quad (\lambda y.(\lambda z.zy)(xy))^{\mathsf{db}}$$

Definition 3.6 (α-Equivalence)**.** *Two λ-terms t, u are α-**equivalent**, notation $t =_\alpha u$, if $t^{\mathsf{db}} = u^{\mathsf{db}}$.*

Notation 3.7. *From now on, λ-terms will be always considered up-to α-equivalence. In particular, we still write Λ for the set Λ quotiented by $=_\alpha$.*

We refer to [Cha12] for further details and references on the locally nameless representation. For other approaches and references see *e.g.* [SU06, Chap. 1] or [Kri93, Chap. 1].

3.3. Beta-Reduction

Beta-reduction (written β-reduction) is the core computation mechanism of the λ-calculus. The basic idea of β-reduction can be phrased as follows:

- $(\lambda x.t)u$ (read "function $\lambda x.t$ applied to argument u") is evaluated by replacing the formal parameter of the function (the bound variable x) by its actual parameter (the argument u).

If we informally write

$$t \rhd_\beta u \qquad \text{for} \qquad t \text{ evaluates to } u$$

then following are expected behaviours of β-reduction.

$$\begin{array}{lclclcl}
\mathtt{id}\;\mathtt{id} & = & (\lambda x.x)\mathtt{id} & \rhd_\beta & \mathtt{id} & & \\
\mathtt{id}\;\delta & = & (\lambda x.x)\delta & \rhd_\beta & \delta & & \\
\delta\;\mathtt{id} & = & (\lambda x.xx)\mathtt{id} & \rhd_\beta & \mathtt{id}\;\mathtt{id} & \rhd_\beta & \mathtt{id} \\
\Omega = \delta\delta & = & (\lambda x.xx)\delta & \rhd_\beta & \delta\delta & \rhd_\beta & \dots
\end{array}$$

3.3.1. Capture-Avoiding Substitution

The key technical ingredient of β-reduction is the operation of **capture-avoiding substitution**.

Definition 3.8 (Capture-Avoiding Substitution). *The* ***capture-avoiding substitution*** $t[u/x]$ *is defined by induction on* t *as follows:*

$$\begin{array}{rcll} y[u/x] & := & y & \textit{if } y \neq x \\ x[u/x] & := & u & \\ (t_1\, t_2)[u/x] & := & t_1[u/x]\, t_2[u/x] & \\ (\lambda y.t)[u/x] & := & \lambda y.t[u/x] & \textit{if } y \neq x \textit{ and } y \notin \mathrm{FV}(u) \end{array}$$

Remark 3.9. *Capture avoiding substitution is a total operation on* λ*-terms quotiented by* α*-equivalence.*

3.3.2. The Notion of Beta-Reduction

Definition 3.10 (Notion of β-reduction). *The* ***notion of*** β***-reduction*** *is the relation* $\rhd_\beta$ *defined as*

$$(\lambda x.t)u \quad \rhd_\beta \quad t[u/x]$$

Terms of the form $(\lambda x.t)u$ *are called* β***-redexes****.*

3.3.3. The Relation of Beta-Reduction

The notion of β-reduction is actually too weak, since we may have to reduce β-redexes which are not at top level. Consider for instance the terms:

$$\mathtt{id}(\lambda xy.yx)(\lambda x.yx) \qquad \lambda x.(\lambda y.xx)\mathtt{id} \qquad ((\lambda x.xx)\mathtt{id})\mathtt{id}$$

Definition 3.11 (Congruence). *Let* $\mathcal{R}$ *be a binary relation on* λ*-terms.*

(1) We say that $\mathcal{R}$ *is* ***compatible*** *if it is closed under the rules*

$$\frac{t \;\; \mathcal{R} \;\; t'}{t\,u \;\; \mathcal{R} \;\; t'\,u} \qquad \frac{u \;\; \mathcal{R} \;\; u'}{t\,u \;\; \mathcal{R} \;\; t\,u'} \qquad \frac{t \;\; \mathcal{R} \;\; t'}{\lambda x.t \;\; \mathcal{R} \;\; \lambda x.t'}$$

(2) We say that $\mathcal{R}$ *is a* ***congruence*** *if* $\mathcal{R}$ *is a compatible equivalence relation.*

Definition 3.12 (Full (Strong) β-Reduction). *The relation of full (strong)* β*-reduction (still denoted* $\rhd_\beta$*) is the least compatible relation such that* $(\lambda x.t)u \;\; \rhd_\beta \;\; t[u/x]$*.*

Remark 3.13. *Note that* α*-conversion may need to be performed on the fly in a sequence of* β*-reductions:*

$$\mathtt{id}(\lambda xy.yx)(\lambda x.yx) \quad \rhd_\beta \quad (\lambda xy.yx)(\lambda x.yx) \quad =_\alpha \quad (\lambda xz.zx)(\lambda x.yx) \quad \rhd_\beta \quad \lambda z.z(\lambda x.yx)$$

Recall that Λ_0 stands for the set of closed λ-terms (Def. 3.4, §3.2.3).

Remark 3.14. *Note that* Λ_0 *is stable under* β*-reduction: if* $t \in \Lambda_0$ *and* $t \rhd_\beta u$ *then* $u \in \Lambda_0$*.*

3.4. Reductions, Conversions and Confluence

Let us look at some possibilities allowed by β-reduction.

Example 3.15. *Recall the λ-terms from Ex. 3.1 (§3.2.2).*

(1) Given $n \in \mathbb{N}$ we have

$$\underline{n}\ \mathtt{id}\ \mathtt{id} \quad \rhd^2_\beta \quad \underbrace{\mathtt{id}(\dots(\mathtt{id}}_{n}\,\mathtt{id})\dots) \quad \rhd^n_\beta \quad \mathtt{id}$$

*So we may have to consider **many** consecutive steps of β-reduction.*

(2) We have two ways for reducing (id id)(id id) *to* id*, namely*

$$\begin{array}{lllllll}
(\mathtt{id}\ \mathtt{id})(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & \mathtt{id}(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & \mathtt{id}\ \mathtt{id} & \rhd_\beta & \mathtt{id} \\
(\mathtt{id}\ \mathtt{id})(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & (\mathtt{id}\ \mathtt{id})\mathtt{id} & \rhd_\beta & \mathtt{id}\ \mathtt{id} & \rhd_\beta & \mathtt{id}
\end{array}$$

This gives three ways for reducing δ (id id) *to* id id*:*

$$\begin{array}{lllllllll}
\delta\ (\mathtt{id}\ \mathtt{id}) & \rhd_\beta & (\mathtt{id}\ \mathtt{id})(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & \mathtt{id}(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & \mathtt{id}\ \mathtt{id} & \rhd_\beta & \mathtt{id} \\
\delta\ (\mathtt{id}\ \mathtt{id}) & \rhd_\beta & (\mathtt{id}\ \mathtt{id})(\mathtt{id}\ \mathtt{id}) & \rhd_\beta & (\mathtt{id}\ \mathtt{id})\mathtt{id} & \rhd_\beta & \mathtt{id}\ \mathtt{id} & \rhd_\beta & \mathtt{id}
\end{array}$$

and

$$\delta\ (\mathtt{id}\ \mathtt{id}) \quad \rhd_\beta \quad \delta\ \mathtt{id} \quad \rhd_\beta \quad \mathtt{id}\ \mathtt{id} \quad \rhd_\beta \quad \mathtt{id}$$

Our goal in this §3.4 is to introduce notations for iterated β-reduction and to discuss a simple (but not trivial) property of β-reduction on pure λ-terms, called **confluence**, which says any two different (finite) iterations of β-reduction from a given λ-term can always be joined by (finite) iterations of β-reduction (see Thm. 3.20 and Def. 3.19 below).

On our way we shall note some important well-known facts.

We begin with the following (usual) definitions, for which a good reference is the textbook [BN98].

Definition 3.16 (Abstract Reduction System)**.** *An **ARS** $(A, \rhd)$ is set A equipped with a binary relation $\rhd \subseteq A \times A$.*

Let $(A, \rhd)$ be an ARS.

- The **transitive closure** of $\rhd$, written $\rhd^+$, is the least relation closed under the rules:

$$\frac{a \ \rhd\ b}{a\ \rhd^+\ b} \qquad\qquad \frac{a\ \rhd^+\ b \qquad b\ \rhd^+\ c}{a\ \rhd^+\ c}$$

- The **reflexive and transitive closure** of $\rhd$, written $\rhd^*$, is the least relation closed under the rules:

$$\frac{a\ \rhd\ b}{a\ \rhd^*\ b} \qquad\qquad \frac{}{a\ \rhd^*\ a} \qquad\qquad \frac{a\ \rhd^*\ b \qquad b\ \rhd^*\ c}{a\ \rhd^*\ c}$$

- The **symmetric, reflexive and transitive closure** of $\rhd$, written $\lhd\rhd^*$, is the least relation closed under the rules:

$$\frac{a \ \rhd \ b}{a \ \lhd\rhd^* \ b} \qquad \frac{}{a \ \lhd\rhd^* \ a} \qquad \frac{a \ \lhd\rhd^* \ b}{b \ \lhd\rhd^* \ a} \qquad \frac{a \ \lhd\rhd^* \ b \qquad b \ \lhd\rhd^* \ c}{a \ \lhd\rhd^* \ c}$$

Notation 3.17. *In the case of β-reduction, we write $t =_\beta u$ for $t \lhd\rhd^*_\beta u$.*

Example 3.18. *For instance, we have*

$$\begin{array}{rcl}
\mathtt{id}\ \mathtt{id} & \rhd^+_\beta & \mathtt{id} \\
\delta\ \mathtt{id} & \rhd^+_\beta & \mathtt{id} \\
\delta\ \delta & \rhd^+_\beta & \delta\ \delta \\
\mathtt{id} & \rhd^*_\beta & \mathtt{id} \\
(\mathtt{id}\ \mathtt{id})\mathtt{id} & =_\beta & \mathtt{id}(\mathtt{id}\ \mathtt{id})
\end{array}$$

Definition 3.19 (Confluence)**.** *Let $(A, \rhd)$ be an ARS.*

- *The relation $\rhd$ is **confluent** if whenever*

$$a \rhd^* b \quad \textit{and} \quad a \rhd^* c$$

there is $d \in A$ such that

$$b \rhd^* d \quad \textit{and} \quad c \rhd^* d$$

- *The relation $\rhd$ is **locally confluent** if whenever*

$$a \rhd b \quad \textit{and} \quad a \rhd c$$

there is $d \in A$ such that

$$b \rhd^* d \quad \textit{and} \quad c \rhd^* d$$

- *The relation $\rhd$ has the **diamond property** if whenever*

$$a \rhd b \quad \textit{and} \quad a \rhd c$$

there is $d \in A$ such that

$$b \rhd d \quad \textit{and} \quad c \rhd d$$

We can now state Thm. 3.20.

Theorem 3.20 (Confluence of β-reduction)**.** *The relation $\rhd_\beta$ is confluent.*

We shall prove Thm. 3.20 in §3.5.

A important consequence of Thm. 3.20 is that the equational theory $=_\beta$ is consistent, in the sense that there terms t, u such that $t \neq_\beta u$. This is given by the following simple and well-known property. See *e.g.* [BN98, Thm. 2.1.5] for a proof.

Proposition 3.21 (Church-Rosser and Confluence)**.** *Let $(A, \rhd)$ be an ARS. Then $\rhd$ is confluent if and only if whenever*

$$a \quad \lhd\rhd^* \quad b$$

there is $c \in A$ such that

$$a \rhd^* c \quad \text{and} \quad b \rhd^* c$$

Corollary 3.22. *The equational theory $=_\beta$ is consistent.*

Proof. Exercise! □

Remark 3.23. *Note that if $\mathcal{R}$ is congruence (Def. 3.11, §3.3.3) which contains $=_\beta$ and such that $\mathtt{T}\,\mathcal{R}\,\mathtt{F}$, then we have $t\,\mathcal{R}\,u$ for all λ-terms t, u.*

Proof. Exercise! □

3.4.1. Undecidability of Beta-Conversion

We mention here some important well-known results on the undecidability of β-conversion for the (pure) untyped λ-calculus. Recall that Λ_0 stands for the set of closed λ-terms (Def. 3.4, §3.2.3).

Definition 3.24. *The relation of **closed** β-conversion $=_{\beta_0}$ is the symmetric reflexive transitive closure of $(\Lambda_0, \rhd_\beta)$.*

Note that it follows from the confluence of β-reduction (Thm. 3.20) together with Prop. 3.21 and Rem. 3.14 (§3.3) that given closed λ-terms $t, u \in \Lambda_0$, if $t =_\beta u$ then $t =_{\beta_0} u$.

Definition 3.25.

*(1) A **β-equationally saturated set** is a set $S \subseteq \Lambda$ which is stable under β-conversion (i.e. if $t \in S$ and $t =_\beta u$ then $u \in S$).*

*A β-equationally saturated set S is **non-trivial** if $S \neq \emptyset$ and $S \neq \Lambda$.*

*(2) A **β_0-equationally saturated set** is a set $S_0 \subseteq \Lambda_0$ which is stable under $=_{\beta_0}$.*

*A β_0-equationally saturated set S_0 is **non-trivial** if $S_0 \neq \emptyset$ and $S_0 \neq \Lambda_0$.*

Theorem 3.26.

(1) If S is β-equationally saturated and non-trivial then S is not recursive.

(2) If S_0 is β_0-equationally saturated and non-trivial then S_0 is not recursive.

Theorem 3.26.(1) is [Bar92, Thm. 2.2.15] (see also [Bar84, Thm. 6.6.2]), and Thm. 3.26.(2) is [Bar84, Cor. 6.6.4]. Both results follow from the Turing-completeness of the untyped λ-calculus (see *e.g.* [Bar92, Thm. 2.2.12]) and from manipulation of suitable Gödel numbers (see *e.g.* [Bar92, §2.2]).

We are interested in Thm. 3.26 for its following corollaries.

Corollary 3.27.

(1) The set following set of λ-terms is not recursive:

$$\{t \in \Lambda \mid t =_\beta \mathtt{T}\}$$

(2) The set following set of closed λ-terms is not recursive:

$$\{t \in \Lambda_0 \mid t =_{\beta_0} \mathtt{T}\}$$

Corollary 3.27 is essentially due to Church (see [Bar84, §6.6] for references). The undecidability of β-conversion is an immediate consequence.

Corollary 3.28. *The following problems are undecidable:*

- *Given λ-terms t and u, decide whether $t =_\beta u$.*
- *Given closed λ-terms t and u, decide whether $t =_{\beta_0} u$.*

3.4.2. A Further Simple Fact on Confluence

Returning to Def. 3.19, it is not difficult to show that

$$\text{Diamond Property} \quad \Longrightarrow \quad \text{Confluence} \quad \Longrightarrow \quad \text{Local Confluence}$$

We sketch here the proof of the fact that the diamond property implies confluence, since this plays a crucial role in our proof of Thm. 3.20.

Notation 3.29. *Given an ARS $(A, \rhd)$ and $n \in \mathbb{N}$, let $a \rhd^n b$ if there are $c_0, \dots, c_n \in A$ such that $a = c_0$, $b = c_n$ and $c_i \rhd c_{i+1}$ for all $i < n$.*

Lemma 3.30. *Let $(A, \rhd)$ be an ARS. Assume that $\rhd$ satisfies the diamond property. Let $a, b, c \in A$ and $n, m \in \mathbb{N}$ such that $a \rhd^n b$ and $a \rhd^m c$. Then there is some $d \in A$ such that $b \rhd^m d$ and $c \rhd^n d$.*

Proof. Exercise! □

The confluence of a relation with the diamond property is then given by the following.

Lemma 3.31. *Let $(A, \rhd)$ be an ARS. We have*

$$\rhd^* \quad = \quad \bigcup_{n \in \mathbb{N}} \rhd^n$$

Proof. Exercise! □

Corollary 3.32. *If $(A, \rhd)$ has the diamond property, then $(A, \rhd)$ is confluent.*

3.4.3. Normalization and Newman's Lemma

We briefly discuss here a well-known method to prove the confluence of some ARS's. While this method is not applicable to prove the confluence of the untyped λ-calculus, it is of some use for **typed** extensions of the pure λ-calculus (*e.g.* §4.4).

Definition 3.33 (Normal forms and Normalization)**.** *Let $(A, \rhd)$ be an ARS.*

- *We say that $a \in A$ is a **normal form** if there is no $b \in A$ such that $a \rhd b$.*
- *We say that $a \in A$ is **weakly normalizing** if there is a normal form $b \in B$ such that $a \rhd^* b$. In this case we say that b is a normal form of a.*

  We say that $\rhd$ is weakly normalizing if every $a \in A$ is weakly normalizing.
- *We say that $a \in A$ is **strongly normalizing** if there is no infinite reduction sequence starting from a.*

  We say that $\rhd$ is strongly normalizing if every $a \in A$ is strongly normalizing.

When we just say "normalizing", we usually mean **weak** rather than **strong** normalization.

Example 3.34. *For instance, w.r.t. β-reduction,*

- *the λ-terms* `id` *and* (`id id`) *are strongly normalizing,*
- *the λ-term $(\delta\ \delta)$ is **not** (even weakly) normalizing,*
- *the λ-term $(\lambda xy.y)(\delta\ \delta)$ is not strongly normalizing, but it is weakly normalizing.*

A strongly normalizing ARS is confluent whenever it is locally confluent. See *e.g.* [BN98, Lem. 2.7.2] for a proof.

Lemma 3.35 (Newman)**.** *Let $(A, \rhd)$ be an ARS. If $\rhd$ is locally confluent and strongly normalizing, then $\rhd$ is confluent.*

Note that (a necessarily not strongly normalizing) $(A, \rhd)$ can be locally confluent and (weakly) normalizing without being confluent.

Example 3.36. *The ARS $(\{a, b, c, d\}, \rightarrow)$, where*

$$\begin{tikzcd} a & b \arrow[l] \arrow[r, bend left] & c \arrow[l, bend left] \arrow[r] & d \end{tikzcd}$$

is locally confluent, weakly normalizing but not confluent (since a and d are distinct normal forms).

Remark 3.37. *Newman's lemma cannot be applied to prove the confluence of $\rhd_\beta$ on untyped λ-terms, since not every λ-term is strongly β-normalizing (e.g. $\Omega \rhd_\beta \Omega$).*

3.5. Confluence of Beta-Reduction

We now prove Thm. 3.20. We use a technique due to Tait and Martin-Löf (see *e.g.* [SU06, §1.4]). The main idea is to target a diamond property. Note that $\rhd_\beta$ does not have the diamond property since the two one-step β-reducts of $(\delta(\texttt{id id}))$, namely

$$(\texttt{id id})\ (\texttt{id id}) \qquad \lhd_\beta \qquad \delta\ (\texttt{id id}) \qquad \rhd_\beta \qquad \delta\ \texttt{id}$$

can be joined, but not in one step of β-reduction.

The trick is to look for a relation $\rhd_{\|\beta}$ with the diamond property and such that $\rhd_\beta \subseteq \rhd_{\|\beta} \subseteq \rhd^*_\beta$. The example of $\delta(\texttt{id id})$ above suggests to allow for **parallel** β-reduction steps in $\rhd_{\|\beta}$, Technically, we assume that $\rhd_{\|\beta}$ is reflexive, compatible (Def. 3.11, §3.3.3), and closed under the rule

$$\frac{t \ \rhd_{\|\beta}\ t' \qquad u\ \rhd_{\|\beta}\ u'}{tu\ \rhd_{\|\beta}\ t'u'}$$

We can then close the case of $\delta(\texttt{id id})$ above with

$$(\texttt{id id})(\texttt{id id}) \quad \rhd_{\|\beta} \quad \texttt{id id}$$

Now, allowing for parallel β-reduction leads to the following one-step reducts:

$$\big(\lambda y.(\texttt{id id})\ (\texttt{id id})\big)\texttt{id} \qquad \lhd_\beta \qquad \big(\lambda x.((\lambda y.xx)\texttt{id})\big)(\texttt{id id}) \qquad \rhd_{\|\beta} \qquad \delta\ \texttt{id}$$

The relation $\rhd_{\|\beta}$ is further required to allow for the simultaneous reduction of **nested** β-redexes, as expressed by the rule

$$\frac{t \ \rhd_{\|\beta}\ t' \qquad u\ \rhd_{\|\beta}\ u'}{(\lambda x.t)u\ \rhd_{\|\beta}\ t'[u'/x]}$$

We thus have

$$\big(\lambda y.(\texttt{id id})\ (\texttt{id id})\big)\texttt{id} \quad \rhd_{\|\beta} \quad \texttt{id id}$$

and conclude the example with $\delta\,\texttt{id}\ \rhd_\beta\ \texttt{id id}$.

As we shall see in §3.5.2 below, the above requirements lead to a relation $\rhd_{\|\beta}$ with the diamond property and such that $\rhd_\beta \subseteq \rhd_{\|\beta} \subseteq \rhd^*_\beta$.

3.5.1. Some Basic Properties

We begin with some basic properties of substitution and (usual) β-reduction. First, a simple property of $\rhd^*_\beta$.

Lemma 3.38. *The relation $\rhd^*_\beta$ is closed under the two following rules*

$$\frac{t\ \rhd^*_\beta\ t'}{\lambda x.t\ \rhd^*_\beta\ \lambda x.t'} \qquad \frac{t\ \rhd^*_\beta\ t' \qquad u\ \rhd^*_\beta\ u'}{tu\ \rhd^*_\beta\ t'u'}$$

Proof. Exercise! $\square$

Remark 3.39. *Otherwise said, $\rhd_\beta^*$ is a compatible relation (Def. 3.11, §3.3.3). We similalry obtain that $=_\beta$ is compatible, so that it is actually a congruence.*

We now turn to a standard technical property on iterated substitutions. See *e.g.* [SU06, Lem. 1.2.20] or [Bar92, Lem. 2.3.14].

Lemma 3.40. *If $x \notin \mathrm{FV}(v)$ and $x \neq y$ then*

$$t[u/x][v/y] \quad = \quad t[v/y][u[v/y]/x]$$

Proof. Exercise! $\square$

The following is an easy property of β-reduction.

Lemma 3.41.

(1) If $t \rhd_\beta u$ then $t[v/x] \rhd_\beta u[v/x]$.

(2) If $t \rhd_\beta^* u$ then $t[v/x] \rhd_\beta^* u[v/x]$ (and similarly for $\rhd_\beta^+$).

(3) If $t \rhd_\beta u$ then $v[t/x] \rhd_\beta^* v[u/x]$.

Proof. Exercise! $\square$

3.5.2. Parallel Nested Beta-Reduction

We shall prove Theorem 3.20 using an auxiliary notion of **parallel** and **nested** β-reduction.

Definition 3.42 (Parallel Nested β-Reduction)**.** *The relation $\rhd_{\parallel\beta}$ is the least binary relation on λ-terms which is closed under the following rules*

$$\frac{}{x \;\rhd_{\parallel\beta}\; x} \qquad \frac{t \;\rhd_{\parallel\beta}\; t' \qquad u \;\rhd_{\parallel\beta}\; u'}{tu \;\rhd_{\parallel\beta}\; t'u'} \qquad \frac{t \;\rhd_{\parallel\beta}\; t'}{\lambda x.t \;\rhd_{\parallel\beta}\; \lambda x.t'} \qquad \frac{t \;\rhd_{\parallel\beta}\; t' \qquad u \;\rhd_{\parallel\beta}\; u'}{(\lambda x.t)u \;\rhd_{\parallel\beta}\; t'[u'/x]}$$

We begin with some easy basic properties of $\rhd_{\parallel\beta}$.

Lemma 3.43. *The relation $\rhd_{\parallel\beta}$ is reflexive (i.e. $t \rhd_{\parallel\beta} t$ for all λ-terms t).*

Proof. Exercise! $\square$

Lemma 3.44. *We have $\rhd_\beta \subseteq \rhd_{\parallel\beta}$ (i.e. if $t \rhd_\beta u$ then $t \rhd_{\parallel\beta} u$).*

Proof. Exercise! $\square$

Lemma 3.45. *We have $\rhd_{\parallel\beta} \subseteq \rhd_\beta^*$ (i.e. if $t \rhd_{\parallel\beta} u$ then $t \rhd_\beta^* u$).*

Proof. Exercise! $\square$

The key property of $\rhd_{\parallel\beta}$ is the following.

Lemma 3.46. *If $t \rhd_{\parallel\beta} u$ and $v \rhd_{\parallel\beta} w$ then $t[v/x] \rhd_{\parallel\beta} u[w/x]$.*

Proof. Exercise! □

The main interest of $\rhd_{\|\beta}$ is that it satisfies the diamond property.

Proposition 3.47. *The relation $\rhd_{\|\beta}$ satisfies the diamond property.*

Proof. Exercise! □

3.5.3. Proof of Confluence

We can finally prove that $\rhd_\beta$ is confluent (Thm. 3.20).

Proof. Exercise! □

3.5.4. An Alternative Proof by Complete Developments

We briefly discuss here the proof of Prop. 3.47 presented in [SU06] (namely [SU06, Def. 1.4.3 & Lem. 1.4.4]), which relies on the notion of **complete development**. The idea is to reduce in one step all the β-redexes present in a term.

Definition 3.48. *The **complete development** of t, written $t^\bullet$ is inductively defined as:*

$$\begin{array}{rcll} x^\bullet & := & x & \\ (\lambda x.t)^\bullet & := & \lambda x.t^\bullet & \\ (tu)^\bullet & := & t^\bullet u^\bullet & \text{if } t \text{ is not an abstraction} \\ ((\lambda x.t)u)^\bullet & := & t^\bullet[u^\bullet/x] & \end{array}$$

Lemma 3.49. *If $t \rhd_{\|\beta} v$ then $v \rhd_{\|\beta} t^\bullet$.*

Proof. Exercise! □

Corollary 3.50. *The relation $\rhd_{\|\beta}$ satisfies the diamond property.*

Proof. Exercise! □

3.6. Eta-Conversion and Böhm's Theorem

We briefly discuss here a syntactic notion of extensionality for the λ-calculus. More precisely, we are going to give an equational axiomatization of the relation $\cong$ on λ-terms defined as the least equivalence relation closed under the rules

$$\frac{t =_\beta u}{t \cong u} \qquad\qquad \frac{tx \cong ux}{t \cong u}\ (x \notin \mathrm{FV}(t,u)) \qquad\qquad \frac{t \cong t' \qquad u \cong u'}{tu \cong t'u'}$$

The relation $\cong$ can be axiomatized as the least congruence (Def. 3.11, §3.3.3) containing β-reduction and the following notion of **η-reduction**:

$$\frac{}{\lambda x.tx \ \rhd_\eta\ t}\ (x \notin \mathrm{FV}(t))$$

Definition 3.51.

(1) We let $\rhd_{\beta\eta}$ be the least compatible relation containing $\rhd_\beta$ and $\rhd_\eta$.

(2) We let $=_{\beta\eta}$ be the least congruence containing $\rhd_{\beta\eta}$.

Example 3.52. *Given λ-terms t and u, let $t \circ u := \lambda x.t(ux)$. We have*

$$\begin{array}{lclclcl} t \circ \mathtt{id} & = & \lambda x.t(\mathtt{id}\ x) & \rhd_\beta & \lambda x.tx & \rhd_\eta & t \\ \mathtt{id} \circ t & = & \lambda x.\mathtt{id}(t\ x) & \rhd_\beta & \lambda x.tx & \rhd_\eta & t \end{array}$$

We shall now see that the relation $\cong$ is precisely $=_{\beta\eta}$. See [SU06, Prop. 1.3.10] (also [Bar84, Prop. 3.3.2]). We first make the following simple observation.

Lemma 3.53. *If $t \cong u$ then $\lambda x.t \cong \lambda x.u$.*

PROOF. Exercise! □

Proposition 3.54. *Given λ-terms t, u, the following are equivalent:*

(i) $t \cong u$

(ii) $t =_{\beta\eta} u$

PROOF. Exercise! □

The reduction relation $\rhd_{\beta\eta}$ is confluent, see [Kri93, Thm. 1.30] or [Bar84, Thm. 3.3.9].

Proposition 3.55. *The relation $\rhd_{\beta\eta}$ is confluent.*

In particular, the equational theory $=_{\beta\eta}$ is consistent (via Prop. 3.21). A striking property of $=_{\beta\eta}$ is that it is a **maximal** consistent equational theory on **normalizable** λ-terms. See [Bar84, Cor. 10.4.3] or [Kri93, §5].

Theorem 3.56 (Böhm). *Let t, u be two $\beta\eta$-normal forms such that $t \neq_{\beta\eta} u$. Let $\cong_{(t,u)}$ be the least congruence containing $=_{\beta\eta}$ and such that $t \cong_{(t,u)} u$. Then $\cong_{(t,u)}$ is inconsistent, i.e. $v \cong_{(t,u)} w$ for all λ-terms v, w.*

Note that Thm. 3.56 still holds if t and u are only assumed to be $\beta\eta$-normalizable.

Remark 3.57. *The assumption that t and u are $\beta\eta$-normal(izable) is essential for Thm. 3.56 since there are consistent extensions of $=_{\beta\eta}$ on non-normalizable λ-terms. See [Bar84, Chap. 4].*

Remark 3.58. *The (restriction of the) relation $=_{\beta\eta}$ on typed λ-terms (§4) is essential for denotational semantics.*[4]

[4] See *e.g.* [AC98, §4.2–4.6] (outside the scope of this course).

3.7. Combinatory Logic

We briefly discuss a presentation of **Combinatory Logic**, a simple framework with no primitive notion of λ-abstraction, but in which λ-abstractions can be represented.

The terms of combinatory logic are given by

$$t, u \in \mathcal{C} \quad ::= \quad x \quad | \quad t\ u \quad | \quad \mathtt{s} \quad | \quad \mathtt{k}$$

where $x \in \mathcal{X}$ (§3.2.1). Note that the **closed** terms of combinatory logic are given by

$$t, u \in \mathcal{C}_0 \quad ::= \quad t\ u \quad | \quad \mathtt{s} \quad | \quad \mathtt{k}$$

We write $t[u/x]$ for the substitution of x by u in t. The **conversion relation** $=_{\mathrm{sk}}$ is the least binary relation on $\mathcal{C}$ which is closed under the following rules:

$$\frac{}{t =_{\mathrm{sk}} t} \qquad \frac{t =_{\mathrm{sk}} u}{u =_{\mathrm{sk}} t} \qquad \frac{t =_{\mathrm{sk}} u \qquad u =_{\mathrm{sk}} v}{t =_{\mathrm{sk}} v}$$

$$\frac{t =_{\mathrm{sk}} t' \qquad u =_{\mathrm{sk}} u'}{t\ u =_{\mathrm{sk}} t'\ u'} \qquad \frac{}{\mathtt{k}\ t\ u =_{\mathrm{sk}} t} \qquad \frac{}{\mathtt{s}\ t\ u\ v =_{\mathrm{sk}} t\ v\ (u\ v)}$$

We write $=_{\mathrm{sk}_0}$ for the least relation on $\mathcal{C}_0$ closed under the above rules.

Lemma 3.59. *For each term t, we have $(\mathtt{s}\ \mathtt{k}\ \mathtt{k}\ t) =_{\mathrm{sk}} t$. Moreover, $(\mathtt{s}\ \mathtt{k}\ \mathtt{k}\ t) =_{\mathrm{sk}_0} t$ if t is closed.*

We now discuss a coding of lambda-abstractions using the combinators $\mathtt{s}$ and $\mathtt{k}$. Given a term $t \in \mathcal{C}$ and a variable x, define the term $\boldsymbol{\lambda} x.t \in \mathcal{C}$ by induction on t:

$$\begin{array}{rcll}
\boldsymbol{\lambda} x.x & := & \mathtt{s}\ \mathtt{k}\ \mathtt{k} & \\
\boldsymbol{\lambda} x.t & := & \mathtt{k}\ t & \text{if } x \text{ does not occur in } t \\
\boldsymbol{\lambda} x.(tu) & := & \mathtt{s}\ (\boldsymbol{\lambda} x.t)\ (\boldsymbol{\lambda} x.u) &
\end{array}$$

Note that x does **not** occur in $\boldsymbol{\lambda} x.t$. Moreover, $\boldsymbol{\lambda} x.t$ is closed if t has no other variable than x.

Remark 3.60. *The above definition of $\boldsymbol{\lambda} x.t$ is consistent with [Bar84, Def. 7.1.5]. A possible variant of $\boldsymbol{\lambda} x.t$ is given by the following (see [Bar84, Def. 7.3.4]):*

$$\begin{array}{rcll}
\boldsymbol{\lambda} x.x & := & \mathtt{s}\ \mathtt{k}\ \mathtt{k} & \\
\boldsymbol{\lambda} x.y & := & \mathtt{k}\ y & \textit{if } y \neq x \\
\boldsymbol{\lambda} x.t & := & \mathtt{k}\ t & \textit{if } t \in \{\mathtt{s}, \mathtt{k}\} \\
\boldsymbol{\lambda} x.(tu) & := & \mathtt{s}\ (\boldsymbol{\lambda} x.t)\ (\boldsymbol{\lambda} x.u) &
\end{array}$$

Lemma 3.61. *For all terms t and u, we have $(\boldsymbol{\lambda} x.t)u =_{\mathrm{sk}} t[u/x]$. Moreover, $(\boldsymbol{\lambda} x.t)u =_{\mathrm{sk}_0} t[u/x]$ if t and u are closed.*

Proof. Exercise! □

$$\frac{}{\mathtt{k} \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\mathtt{k}\,x\,y} \qquad \frac{}{\mathtt{k} \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.x}$$

$$\frac{}{\mathtt{s} \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\boldsymbol{\lambda} z.\mathtt{s}\,x\,y\,z} \qquad \frac{}{\mathtt{s} \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\boldsymbol{\lambda} z.xy(yz)}$$

$$\frac{}{\boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\mathtt{s}(\mathtt{k}\,x)(\mathtt{k}\,y) \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\mathtt{k}(xy)}$$

$$\frac{}{\boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\mathtt{s}(\mathtt{s}(\mathtt{kk})x)y \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\boldsymbol{\lambda} z.xz}$$

$$\frac{}{\boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\boldsymbol{\lambda} z.\mathtt{s}(\mathtt{s}(\mathtt{s}(\mathtt{ks})x)y)z \;=_{\mathrm{sk}\beta}\; \boldsymbol{\lambda} x.\boldsymbol{\lambda} y.\boldsymbol{\lambda} z.\mathtt{s}(\mathtt{s}\,x\,z)(\mathtt{s}\,y\,z)}$$

Figure 3: Additional Conversion Rules Combinatory Logic.

3.7.1. Representation of Beta-Conversion

In view of Lem. 3.61, it makes sense to consider the following translation $\mathcal{C}(-)$ of λ-terms (see [Bar84, Def. 7.3.1]):

$$\begin{array}{lcl} \mathcal{C}(x) & := & x \\ \mathcal{C}(t\,u) & := & \mathcal{C}(t)\;\mathcal{C}(u) \\ \mathcal{C}(\lambda x.t) & := & \boldsymbol{\lambda} x.\mathcal{C}(t) \end{array}$$

Note that $\mathcal{C}(t)$ is closed if t is a closed λ-term.

We would have liked to have $\mathcal{C}(t) =_{\mathrm{sk}} \mathcal{C}(u)$ whenever $t =_\beta u$, but this unfortunately does not hold, essentially because $=_{\mathrm{sk}}$ is too weak to be stable under the rule

$$\frac{t =_{\mathrm{sk}} u}{\boldsymbol{\lambda} x.t =_{\mathrm{sk}} \boldsymbol{\lambda} x.u}$$

Example 3.62. *Consider the λ-term $t := \lambda x.\mathtt{id}\ x$. We obviously have $t =_\beta \mathtt{id}$, but we* ***dot not*** *have $\mathcal{C}(t) =_{\mathrm{sk}} \mathcal{C}(\mathtt{id})$. First, note that*

$$\mathcal{C}(\mathtt{id}) \quad = \quad \boldsymbol{\lambda} x.x \quad = \quad \mathtt{s\ k\ k}$$

so that

$$\mathcal{C}(t) \quad = \quad \boldsymbol{\lambda} x.(\mathtt{s\ k\ k})x \quad = \quad \mathtt{s(k(s\ k\ k))(s\ k\ k)}$$

However it can be shown that $\mathtt{s(k(s\ k\ k))(s\ k\ k)} \neq_{\mathrm{sk}} \mathtt{s\ k\ k}$. This follows from Prop. 3.21 (§3.4) and the confluence of the compatible closure (in $\mathcal{C}$) of the relation $\rhd_{\mathrm{sk}}$ given by

$$\mathtt{s}\ t\ u\ v \quad \rhd_{\mathrm{sk}} \quad t\ v\ (u\ v) \qquad \text{and} \qquad \mathtt{k}\ t\ u \quad \rhd_{\mathrm{sk}} \quad t$$

(see e.g. [Bar84, 7.2.4]).

The following is [Bar84, Thm. 7.3.10], where $=_{\mathrm{sk}\beta}$ (resp. $=_{\mathrm{sk}\beta_0}$) is the extension of $=_{\mathrm{sk}}$ (resp. of $=_{\mathrm{sk}_0}$) with the rules of Fig. 3 (which actually only consist of closed terms).

Theorem 3.63. *Given* ***λ-terms*** *t and u, we have $t =_\beta u$ if and only if $\mathcal{C}(t) =_{\mathrm{sk}\beta} \mathcal{C}(u)$. If t and u are closed, then $t =_{\beta_0} u$ if and only if $\mathcal{C}(t) =_{\mathrm{sk}\beta_0} \mathcal{C}(u)$.*

Remark 3.64. *Actually [Bar84, Thm. 7.3.10], only gives the part of Thm. 3.63 on open λ-terms. Consider the case of $t, u \in \Lambda_0$ with $t =_{\beta_0} u$. Then we obviously have $t =_\beta u$ and thus $\mathcal{C}(t) =_{\mathrm{sk}\beta} \mathcal{C}(u)$. Now, since $=_{\mathrm{sk}\beta}$ is closed under substitution, if the derivation of $\mathcal{C}(t) =_{\mathrm{sk}\beta} \mathcal{C}(u)$ involves some variables, then they can be substituted by closed terms, and we indeed get $\mathcal{C}(t) =_{\mathrm{sk}\beta_0} \mathcal{C}(u)$.*

Our motivation for Thm. 3.63 is the following consequence of Cor. 3.28 (§3.4.1):

Corollary 3.65. *The following problems are undecidable:*

- *Given $t, u \in \mathcal{C}$, decide whether $t =_{\mathrm{sk}\beta} u$.*
- *Given $t, u \in \mathcal{C}_0$, decide whether $t =_{\mathrm{sk}\beta_0} u$.*

We shall see in §6.2.4 (Thm. 6.35) that Cor. 3.65 gives a very simple proof of undecidability of (intuitionistic) first-order logic.

4. Curry-Howard Correspondence for Intuitionistic Propositional Logic

4.1. Introduction

This Section presents the **Curry-Howard correspondence** for intuitionistic propositional logic:

Logic		Prog. Languages
Formulae	≡	Types
Proofs	≡	Programs

We proceed in two steps. We present in §4.3 the Curry-Howard correspondence between natural deduction for intuitionistic **minimal** implicational logic (§2.4) and the **simply-typed λ-calculus** (STLC) (to be defined in §4.2). The simply-typed λ-calculus is a restriction on the formation of λ-terms (§3.2) based on a notion of **simple types**, which actually precisely amounts to:

Intuitionistic Min. ⇒-Logic		Simply-Typed λ-Calculus
Formulae	≡	Simple Types
Proofs	≡	Typed λ-Terms

We then consider in §4.4 the case of natural deduction for full intuitionistic propositional logic (NJ_0). This amounts to extend the simply-typed λ-calculus with **product** and **sum** types.

Fundamental properties of intuitionistic logic (Thm. 2.9, §2.1) will be inferred from **normalization** properties of the simply-typed λ-calculus (in the sense of §3.4.3), which shall be discussed in §5.

On technical matters, besides [SU06, §3 & §4], we refer to [Gal95] and [GLT89, §3]. Again, beware that we should consistently follow the notations and technical definitions of **none** of these sources.

We refer to [SU06, §4.8] and [Gal95] for historical aspects. Let us just mention that an **informal** conceptual prequel of the Curry-Howard correspondence is the **Brouwer-Heyting-Kolmogorov** (BHK) interpretation of **proofs** of intuitionistic propositional logic. The BHK interpretation may be formulated as follows (where the term "*proof*" should be understood as some informal notion of "witness of evidence"):

- a "*proof*" of $A_1 \wedge A_2$ is a pair of a "*proof*" of A_1 and a "*proof*" of A_2;
- a "*proof*" of $A_1 \vee A_2$ is a (dependent) pair of an $i \in \{1,2\}$ and a "*proof*" of A_i;
- a "*proof*" of $A \Rightarrow B$ is a "function" taking a "*proof*" of A to a "*proof*" of B;
- there is no "*proof*" of $\bot$.

In particular, the BHK interpretation does not specify what should be understood as a "function" in the case of $A \Rightarrow B$. We refer to [SU06, §2.1] and [TvD88a, Chap. 1, §3.1] for more on the BHK interpretation.

4.2. The Simply-Typed Lambda-Calculus

We introduce here a restriction on the formation of λ-terms based on a notion of **typing** (or **typing discipline**). There are **a lot** of notions of typing for the λ-calculus. Besides [SU06], a good synthetic but general account is [Bar92], while [Pie02] is a gentle but comprehensive reference from the point of view of programming languages.[5]

We begin with the simplest notion, namely the (Curry-style) **simply-typed λ-calculus**, which is at the basis (at least conceptually) of most of typing disciplines for λ-terms.

We shall show in §5 that the simply-typed λ-calculus is strongly normalizing (recall from Ex. 3.34 (§3.4.3) that not every λ-term is normalizing). Under the Curry-Howard correspondence, the **types** of the simply-typed λ-calculus will be identified with the **formulae** of (intuitionistic) minimal implicational logic (§2.4); **typed λ-terms** will then correspond to (natural deduction) intuitionistic **proofs** (see §4.3).

Good technical references are [SU06, §4] and [Bar92, §3] (yet again with different notations and technical choices).

4.2.1. Syntax

We consider the following grammar of **simple** (function) **types**:

$$U, T, V \quad ::= \quad \kappa \quad | \quad U \to T$$

where κ ranges over some (unspecified) set of **base types**.

[5]An other interesting reference (but mostly outside the scope of this course) is [AC98].

$$(\textsc{Var})\ \frac{}{\mathcal{E} \vdash x : T}\ ((x:T) \in \mathcal{E})$$

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x.t : U \to T} \qquad (\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash t\,u : T}$$

Figure 4: Typing Rules of the Simply-Typed λ-Calculus.

Notation 4.1. *Similarly as for formulae (Notation 2.2, §2), we assume that the arrow type constructor $\to$ associates to the right, so that $V \to U \to T$ stands for $V \to (U \to T)$ (which is **different** from $(V \to U) \to T$).*

Simple types will be used to restrict the formation of λ-terms (§3.2). Recall that the latter are given by the grammar:

$$t, u \in \Lambda \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad tu$$

where x ranges over the set $\mathcal{X}$ of variables.

A **typing context** (or **typing environment**) is a **list** $\mathcal{E}$ of the form $x_1 : U_1, \dots, x_n : U_n$ where $x_1, \dots, x_n \in \mathcal{X}$ and such that $x_i \neq x_j$ whenever $i \neq j$. When $\mathcal{E} = x_1 : U_1, \dots, x_n : U_n$, we sometimes write $\text{dom}(\mathcal{E})$ for the set $\{x_1, \dots, x_n\}$. The notation $\mathcal{E}, x : U$ always assume $x \notin \text{dom}(\mathcal{E})$.

A **typing judgment** is a triplet of the form

$$\mathcal{E} \vdash t : T$$

Figure 4 lists the **typing rules** of the **simply-typed λ-calculus**. We say that:

- t **has type** T **in the context** $\mathcal{E}$ when the judgment $\mathcal{E} \vdash t : T$ is derivable with these rules;
- t **is typable in the context** $\mathcal{E}$ when there exists a type T such that $\mathcal{E} \vdash t : T$ is derivable;
- t **is typable** when there exist some context $\mathcal{E}$ and some type T such that $\mathcal{E} \vdash t : T$ is derivable;
- T **is inhabited** if there is some λ-term t such that $\vdash t : T$.

The following refer to λ-terms defined in Ex. 3.1 (§3.2.2).

Example 4.2.

(1) For each type T we have $\vdash \texttt{id} : T \to T$ with derivation tree:

$$\frac{\overline{x : T \vdash T}}{\vdash \lambda x.x : T \to T}$$

(2) For each types U, T, V, with have $\vdash \texttt{pair} : U \to T \to (U \to T \to V) \to V$ with the following derivation tree (where $\mathcal{E}$ is the context $x : U,\ y : T,\ k : U \to T \to V$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathcal{E}\vdash k:U\to T\to V}\qquad\dfrac{}{\mathcal{E}\vdash x:T}}{\mathcal{E}\vdash kx:T\to V}\qquad\dfrac{}{\mathcal{E}\vdash y:T}}{x:U,\ y:T,\ k:U\to T\to V\vdash kxy:V}}{x:U,\ y:T\vdash\lambda k.\ kxy:(U\to T\to V)\to V}}{x:U\vdash\lambda y.\lambda k.\ kxy:T\to(U\to T\to V)\to V}}{\vdash\lambda x.\lambda y.\lambda k.\ kxy:U\to T\to(U\to T\to V)\to V}$$

Exercise 4.3. *Derive the following typing judgments:*

(1) $\vdash \texttt{T} : U \to T \to U$

(2) $\vdash \texttt{F} : U \to T \to T$

(3) $\vdash \texttt{S} : \texttt{nat}[T] \to \texttt{nat}[T]$, *where* $\texttt{nat}[T] = T \to (T \to T) \to T$.

Exercise 4.4. *Show that for each $n \in \mathbb{N}$, we have $\vdash \underline{n} : \texttt{nat}[T]$.*

4.2.2. Main Properties

We gather here some important basic properties of the simply-typed λ-calculus. See [SU06, §3] for more.[6] In the following, the notation $\rhd_\beta$ refers to the full relation of β-reduction (Def. 3.12, §3.3.3).

We begin with simple basic properties, which are proved by induction on derivations.

Fact 4.5 (Inversion)**.**

(1) If $\mathcal{E} \vdash x : T$, then $(x : T) \in \mathcal{E}$.

(2) If $\mathcal{E} \vdash \lambda x.t : T$, then T is of the form $V \to U$ and $\mathcal{E}, x : V \vdash t : U$.

(3) If $\mathcal{E} \vdash tu : T$ then there is some U such that $\mathcal{E} \vdash t : U \to T$ and $\mathcal{E} \vdash u : U$.

Exercise 4.6. *Show that δ (and thus Ω) is not typable.*

Lemma 4.7 (Structural Properties)**.**

(Weakening) *If $\mathcal{E} \vdash t : T$ and $x \notin \mathrm{dom}(\mathcal{E})$ then $\mathcal{E}, x : U \vdash t : T$.*

(Contraction) *If $\mathcal{E}, x : U, y : U \vdash t : T$ then $\mathcal{E}, x : U \vdash t[x/y] : T$.*

(Exchange) *If $\mathcal{E}, x : U, \mathcal{E}', y : V, \mathcal{E}'' \vdash t : T$, then $\mathcal{E}, y : V, \mathcal{E}', x : U, \mathcal{E}'' \vdash t : T$.*

Example 4.8. *Assuming $\mathcal{E} \vdash t : U \to T$ and $\mathcal{E} \vdash u : V \to U$, we have $\mathcal{E} \vdash t \circ u : V \to T$ (where $t \circ u := \lambda x.t(ux)$), with derivation tree:*

[6] See also [AC98, §4.5] (beyond the scope of this course).

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{E} \vdash t : U \to T \\ \hdashline \mathcal{E}, x : V \vdash t : U \to T\end{array} \qquad \dfrac{\begin{array}{c}\mathcal{E} \vdash u : V \to U \\ \hdashline \mathcal{E}, x : V \vdash u : V \to U\end{array} \qquad \dfrac{}{\mathcal{E}, x : V \vdash x : V}}{\mathcal{E}, x : V \vdash ux : U}}{\mathcal{E}, x : V \vdash t(ux) : T}}{\mathcal{E} \vdash \lambda x.t(ux) : V \to T}$$

The following two properties are required for essentially all type systems.[7]

Lemma 4.9 (Substitution). *If $\mathcal{E}, x : U \vdash t : T$ and $\mathcal{E} \vdash u : U$ then $\mathcal{E} \vdash t[u/x] : T$.*

Proposition 4.10 (Subject Reduction). *If $\mathcal{E} \vdash t : T$ and $t \rhd_\beta u$ then $\mathcal{E} \vdash u : T$.*

Remark 4.11. *In particular, for a typing context $\mathcal{E}$ and a type T, the relation $\rhd_\beta$ is confluent on the set*

$$|\mathcal{E} \vdash T| \quad := \quad \{t \mid \mathcal{E} \vdash t : T\}$$

The following fundamental property of the simply-typed λ-calculus is admitted until §5 (Thm. 5.32, §5.5). Recall from Ex. 3.34 (§3.4.3) that the (non typable) λ-term Ω is not normalizing for β-reduction.

Theorem 4.12 (Strong Normalization). *If $\mathcal{E} \vdash t : T$ then t is strongly β-normalizing (in the sense of Def. 3.33, §3.4.3).*

Theorem 4.12 is a particular case of Thm. 4.31 (§4.4.4).

Remark 4.13 (Decidability of Typing). *The following are decidable problems:*

(1) Given a type T, a context $\mathcal{E}$ and a λ-term t, decide whether $\mathcal{E} \vdash t : T$.

(2) Given a λ-term t, decide whether t is typable.

Decidability of type inhabitation is discussed in Rem. 4.20 (§4.3) below. We refer to [SU06, §3.2] for details.

4.3. Curry-Howard Correspondence for Intuitionistic Minimal Implicational Logic

Recall that the formulae of minimal implicational logic (§2.4) are given by

$$A, B \quad ::= \quad \mathsf{p} \quad | \quad A \Rightarrow B$$

where p is an atomic proposition.

On the other hand, the simple (function) types of §4.2.1 are given by

$$U, T \quad ::= \quad \kappa \quad | \quad U \to T$$

where κ is a base type.

The **Curry-Howard correspondence** for intuitionistic minimal implicational logic stems from the identification of simple (arrow) types with the formulae of minimal implicational logic (see Tab. 1). Under this identification, we may consider that **simple types are formulae of minimal implicational logic**.

[7]Type systems for variants of STLC with other notions of reduction would require different formulations. A (non basic but) relevant example is [Lev03, Prop. 10, §2].

	Simple Types	$\equiv$	$\Rightarrow$-Formulae	
(base types)	κ	$\equiv$	p	(atomic propositions)
(function types)	$(-) \to (-)$	$\equiv$	$(-) \Rightarrow (-)$	(implications)

Table 1: The Curry-Howard Correspondence: Simple Types and $\Rightarrow$-Formulae.

Notation 4.14 (Types as Formulae)**.** *For the remaining of this §4.3 we assume that **types** are given by the grammar of formulae:*

$$A, B \quad ::= \quad \mathsf{p} \quad | \quad A \Rightarrow B$$

This leads to a strong analogy between typing rules on the one hand and deduction rules on the other hand. More precisely, erasing terms (and variables) from the typing rules of Fig. 4 (§4.2.1) exactly gives the deduction rules of §2.4:

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i} \quad \rightsquigarrow \quad \frac{}{A_1, \ldots, A_n \vdash A_i} \quad (\mathrm{Ax})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n, x : A \vdash t : B}{x_1 : A_1, \ldots, x_n : A_n \vdash \lambda x.t : A \Rightarrow B} \quad \rightsquigarrow \quad \frac{A_1, \ldots, A_n, A \vdash B}{A_1, \ldots, A_n \vdash A \Rightarrow B} \quad (\Rightarrow\text{-I})$$

$$\frac{\begin{array}{c} x_1 : A_1, \ldots, x_n : A_n \vdash t : A \Rightarrow B \\ x_1 : A_1, \ldots, x_n : A_n \vdash u : A \end{array}}{x_1 : A_1, \ldots, x_n : A_n \vdash tu : B} \quad \rightsquigarrow \quad \frac{\begin{array}{c} A_1, \ldots, A_n \vdash A \Rightarrow B \\ A_1, \ldots, A_n \vdash A \end{array}}{A_1, \ldots, A_n \vdash B} \quad (\Rightarrow\text{-E})$$

It follows that **typed λ-terms** can be identified with **proofs**:

Intuitionistic Min. $\Rightarrow$-Logic		Simply-Typed λ-Calculus
Formulae	$\equiv$	Simple Types
Proofs	$\equiv$	Typed λ-Terms

In particular, we have the following obvious result, stated as a Theorem for emphasis.

Theorem 4.15 (Curry-Howard Correspondence)**.** *The sequent $A_1, \ldots, A_n \vdash A$ is derivable in intuitionistic minimal implicational logic if and only if there is a λ-term t such that $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$.*

Remark 4.16. *As a direct consequence of Thm. 4.15 (together with Prop. 2.34, §2.4), note that if p, q are distinct atomic propositions, then there is no λ-term t such that $\vdash t : ((\mathsf{p} \Rightarrow \mathsf{q}) \Rightarrow \mathsf{p}) \Rightarrow \mathsf{p}$.*

On the other hand, Thm. 4.15 of course gives λ-terms for all intuitionistic tautologies of minimal implicational logic.

We now consider examples inspired from §2.2. Since we have no proper $\bot$ yet (and thus no proper negation), we replace $\bot$ by a chosen atomic proposition $\mathtt{R}$ and write $\neg_{\mathtt{R}} A$ for $A \Rightarrow \mathtt{R}$. Note that the rule (ExFalso) is unavailable: we are indeed in **minimal logic** (see Rem. 2.16).

Example 4.17. *We have $\lambda a.\lambda k.ka : A \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} A$ with the derivation*

$$\dfrac{\dfrac{\dfrac{}{a : A,\, k : \neg_{\mathtt{R}} A \vdash k : A \Rightarrow \mathtt{R}} \qquad \dfrac{}{x : A,\, k : \neg_{\mathtt{R}} A \vdash a : A}}{a : A,\, k : \neg_{\mathtt{R}} A \vdash ka : \mathtt{R}}}{a : A \vdash \lambda k.ka : \neg_{\mathtt{R}}\neg_{\mathtt{R}} A}$$

Exercise 4.18. *In each case below, give a λ-term t of the prescribed type:*

(1) $\vdash t : \neg_{\mathtt{R}}\neg_{\mathtt{R}}\neg_{\mathtt{R}} A \Rightarrow \neg_{\mathtt{R}} A$

(2) $\vdash t : (A \Rightarrow B) \Rightarrow (\neg_{\mathtt{R}}\neg_{\mathtt{R}} A \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} B)$

(3) $\vdash t : \big((\neg_{\mathtt{R}}\neg_{\mathtt{R}} A \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} B) \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} A\big) \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} A$

In contrast with Rem. 4.16, it is not too difficult to find λ-terms for the Kolmogorov translation of Peirce's law (Rem. 2.43, §2.5).

Exercise 4.19. *Give a λ-term t such that $\vdash t : \neg_{\mathtt{R}}\neg_{\mathtt{R}}\big(\neg_{\mathtt{R}}\neg_{\mathtt{R}}(\neg_{\mathtt{R}}\neg_{\mathtt{R}} A \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} B) \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} A\big) \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} A$.*

Remark 4.20 (Decidability of Type Inhabitation)**.** *Continuing Rem. 4.13 (§4.2), it follows from Thm. 4.15 and the decidability of provability in intuitionistic minimal implicational logic (Rem. 2.36, §2.4) that the following* ***type inhabitation problem*** *is decidable:*

- *Given a type T, decide whether T is inhabited.*

We refer to [SU06, §4.2] for more.

4.3.1. Beta-Reduction and Curry-Howard

Theorem 4.15 extends to a correspondence between **β-reduction** and a notion of computation for natural deduction called **proof normalization**. The idea is that a step of β-reduction

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ \Pi_1\end{matrix}}{\mathcal{E}, x : A \vdash t : B}}{\mathcal{E} \vdash \lambda x.t : A \Rightarrow B} \qquad \dfrac{\begin{matrix}\vdots\\ \Pi_2\end{matrix}}{\mathcal{E} \vdash u : A}}{\mathcal{E} \vdash (\lambda x.t)u : B} \qquad \rhd_\beta \qquad \dfrac{\begin{matrix}\vdots\\ \Pi_1[\Pi_2/x]\end{matrix}}{\mathcal{E} \vdash t[u/x] : B}$$

(where the derivation $\Pi_1[\Pi_2/x]$ is obtained by the Substitution Lemma 4.9, §4.2.2) corresponds to the reduction of **redexes** (⇒-I)/(⇒-E) in natural deduction:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Pi_1}}{\dfrac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B}} \qquad \dfrac{\vdots}{\dfrac{\Pi_2}{\Delta \vdash A}}}{\Delta \vdash B} \qquad \rhd \qquad \dfrac{\vdots}{\dfrac{\Pi_1[\Pi_2/A]}{\Delta \vdash B}}$$

Here, the proof

$$\dfrac{\vdots}{\dfrac{\Pi_1[\Pi_2/A]}{\Delta \vdash B}}$$

stands for the proof Π_1 in which each (Ax) rule

$$\dfrac{}{\Delta' \vdash A}$$

(where Δ is necessarily a prefix of Δ') is replaced by a derivation of $\Delta' \vdash A$ obtained from Π_2 by weakenings (Lem. 2.6, §2.1).

We shall not pursue this direction too far. We refer to [SU06, §4.4], [vD04, §6] and [TS00, §6] for normalization of natural deduction proofs.

4.4. Full Intuitionistic Propositional Logic

We now turn to the Curry-Howard correspondence for the full system NJ_0 of §2.1. This requires an extension of the (simply-typed) λ-calculus that we introduce step by step. We mainly refer to [SU06, §4.5] (see also [Gal95, §3]). See [Pie02, §11] for a programming language perspective on the additional constructs introduced in §4.4.1–§4.4.3 below.

Notation 4.21. *We step back from Notation 4.14 (§4.3) and consider for the moment types and formulae as different entities.*

4.4.1. Intuitionistic Conjunctions and Products

The natural deduction rules for conjunction $(-) \wedge (-)$, namely

$$(\wedge\text{-I})\ \dfrac{\Delta \vdash A \qquad \Delta \vdash B}{\Delta \vdash A \wedge B} \qquad (\wedge_1\text{-E})\ \dfrac{\Delta \vdash A \wedge B}{\Delta \vdash A} \qquad (\wedge_2\text{-E})\ \dfrac{\Delta \vdash A \wedge B}{\Delta \vdash B}$$

suggest to extend the simply-typed λ-calculus with **product types** $(-) \times (-)$ (and the corresponding term formers), as in the following additional typing rules:

$$(\times\text{-I})\ \dfrac{\mathcal{E} \vdash t : T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash \langle t, u\rangle : T \times U} \qquad (\times_1\text{-E})\ \dfrac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_1 t : T} \qquad (\times_2\text{-E})\ \dfrac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_2 t : U}$$

The term former $\langle -, - \rangle$ is usually called **pairing**, while $\pi_1(-), \pi_2(-)$ are **projections**.

Moreover (reversing the approach of §4.3.1), the reduction of $(\wedge\text{-I})/(\wedge_i\text{-E})$-redexes in natural deduction, namely

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Pi_1\end{array}}{\Delta \vdash A_1} \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_2\end{array}}{\Delta \vdash A_2}}{\Delta \vdash A_1 \wedge A_2}}{\Delta \vdash A_i} \qquad \rhd \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_i\end{array}}{\Delta \vdash A_i}$$

suggests the following extension of β-reduction to cope with products and pairs:

$$\begin{array}{lcl} \pi_1\langle t_1, t_2\rangle & \rhd_\beta & t_1 \\ \pi_2\langle t_1, t_2\rangle & \rhd_\beta & t_2 \end{array}$$

Finally, the 0-ary form $\top$ of $(-) \wedge (-)$, with its (unique) natural deduction rule

$$(\top\text{-I})\ \dfrac{}{\Delta \vdash \top}$$

suggests a 0-ary form of $(-) \times (-)$, *i.e.* a **unit type** `unit` with term former $\langle\rangle$ typed as

$$(\texttt{unit}\text{-I})\ \dfrac{}{\mathcal{E} \vdash \langle\rangle : \texttt{unit}}$$

4.4.2. Intuitionistic Disjunctions and Sums

Consider now the natural deduction rules for (intuitionistic) disjunction $(-) \vee (-)$:

$$(\vee_i\text{-I})\ \dfrac{\Delta \vdash A_i}{\Delta \vdash A_1 \vee A_2} \qquad\qquad (\vee\text{-E})\ \dfrac{\Delta \vdash A_1 \vee A_2 \qquad \Delta, A_1 \vdash B \qquad \Delta, A_2 \vdash B}{\Delta \vdash B}$$

These rules correspond closely to the following rules for **sum types** $(-) + (-)$:

$$(+_i\text{-I})\ \dfrac{\mathcal{E} \vdash t : T_i}{\mathcal{E} \vdash \texttt{in}_i\, t : T_1 + T_2} \qquad\qquad (+\text{-E})\ \dfrac{\mathcal{E} \vdash t : T_1 + T_2 \qquad \begin{array}{l}\mathcal{E}, x_1 : T_1 \ \vdash\ u_1 : U \\ \mathcal{E}, x_2 : T_2 \ \vdash\ u_2 : U\end{array}}{\mathcal{E} \vdash \texttt{case}\ t\ \{\texttt{in}_1\, x_1 \mapsto u_1 \mid \texttt{in}_2\, x_2 \mapsto u_2\} : U}$$

The term formers $\texttt{in}_1(-), \texttt{in}_2(-)$ are called **coprojections**.[8] The `case` construction (which comes under various names and notations in different settings) is a close relative of the usual **pattern matching** constructs of (functional) programming languages, while sum types are a form of **variant type** (see [Pie02, §11.8 & §11.9]).

The reduction of $(\vee_i\text{-I})/(\vee\text{-E})$-redexes in natural deduction:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Pi\end{array}}{\Delta \vdash A_i}}{\Delta \vdash A_1 \vee A_2} \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_1\end{array}}{\Delta, A_1 \vdash C} \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_2\end{array}}{\Delta, A_2 \vdash C}}{\Delta \vdash C} \qquad \rhd \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_i[\Pi/A_i]\end{array}}{\Delta \vdash C}$$

[8] An other usual name is "injection" (but it may **wrongly** suggest an injectivity property).

(where $\Pi_i[\Pi/A_i]$ is obtained similarly as in §4.3.1) closely corresponds to the usual reduction rules for pattern matching:

$$\begin{array}{lcl} \texttt{case}\ (\texttt{in}_1\, t)\ \{\texttt{in}_1\, x_1 \mapsto u_1 \mid \texttt{in}_2\, x_2 \mapsto u_2\} & \rhd_\beta & u_1[t/x_1] \\ \texttt{case}\ (\texttt{in}_2\, t)\ \{\texttt{in}_1\, x_1 \mapsto u_1 \mid \texttt{in}_2\, x_2 \mapsto u_2\} & \rhd_\beta & u_2[t/x_2] \end{array}$$

(where that t is of type U_i when $\texttt{in}_i\, t$ is of type $U_1 + U_2$).

Consider now $\bot$, the 0-ary case of $(-) \vee (-)$ with (unique) natural deduction rule

$$(\bot\text{-E})\ \frac{\Delta \vdash \bot}{\Delta \vdash A}$$

This suggests a 0-ary version of $(-) + (-)$, which is provided by the **empty type** `void` with elimination form

$$(\texttt{void}\text{-E})\ \frac{\mathcal{E} \vdash t : \texttt{void}}{\mathcal{E} \vdash \texttt{case}_\bot\ t\ \{\} : T}$$

Note that the (`void`-E)-rule produces a term of **any** type T! On the other hand there is no introduction rule for `void`, so that (in the λ-calculus defined up to now) there is no **closed** term of type `void` (see Cor. 4.34 §4.4.4 below). In particular, the (`void`-E)-rule cannot be applied when the context $\mathcal{E}$ is empty.

Remark 4.22. *Natural deduction is often equipped with stronger reduction rules for (intuitionistic) disjunction than the above, see e.g. [Gal95] and [TS00, §6].*

4.4.3. A Simply-Typed Lambda-Calculus with Sums and Products

To summarize, we end up with a type system for extended λ-terms given by the grammar:

$$\begin{array}{rcl} t, u & ::= & x \mid \lambda x.t \mid tu \mid \langle t, u\rangle \mid \pi_1 t \mid \pi_2 t \\ & \mid & \texttt{in}_1\, t \mid \texttt{in}_2\, t \mid \langle\rangle \mid \texttt{case}_\bot\ t\ \{\} \\ & \mid & \texttt{case}\ t\ \{\texttt{in}_1\, x_1 \mapsto u_1 \mid \texttt{in}_2\, x_2 \mapsto u_2\} \end{array}$$

where $x \in \mathcal{X}$ (as in §3).

Warning 4.23. *The* `case` *construct actually incorporates* ***binders****: the variables x_1, x_2 are* ***bound*** *in* `case` t $\{\texttt{in}_1 x_1 \mapsto u_1 \mid \texttt{in}_2 x_2 \mapsto u_2\}$*. We thus assume the corresponding extension of α-conversion (§3.2.3 and §3.2.4).*

The notions of free-variable (Def. 3.3, §3.2.3) and of capture-avoiding substitution (Def. 3.8, §3.3) are adapted accordingly.

As for reduction, we consider the analogue of Def. 3.12 (§3.3.3) for the extended λ-terms defined above. More precisely, we first define a **basic** reduction relation $\rhd_0$ with the basic rules of Fig. 5. We then let the **full** (**strong**) reduction relation $\rhd_\beta$ be the closure of $\rhd_0$ under the congruence rules of Fig. 5.

Warning 4.24. *What we call β-reduction here is often called differently in the literature, because the reduction rules for products, sums etc. are often given specific names.*

Basic Rules:

$$\begin{array}{lll} (\lambda x.t)u \;\rhd_0\; t[u/x] & & \\ \pi_1\langle t,u\rangle \;\rhd_0\; t & \quad & \texttt{case}\ (\texttt{in}_1\, t)\ \{\texttt{in}_1\, x \mapsto u_1 \mid \texttt{in}_2\, x \mapsto u_2\} \;\rhd_0\; u_1[t/x] \\ \pi_2\langle t,u\rangle \;\rhd_0\; u & \quad & \texttt{case}\ (\texttt{in}_2\, t)\ \{\texttt{in}_1\, x \mapsto u_1 \mid \texttt{in}_2\, x \mapsto u_2\} \;\rhd_0\; u_2[t/x] \end{array}$$

Congruence Rules: $\rhd_\beta$ is the least relation containing $\rhd_0$ and closed under the rules

$$\frac{t \;\rhd_\beta\; t'}{t\,u \;\rhd_\beta\; t'\,u} \qquad \frac{u \;\rhd_\beta\; u'}{t\,u \;\rhd_\beta\; t\,u'} \qquad \frac{t \;\rhd_\beta\; t'}{\lambda x.t \;\rhd_\beta\; \lambda x.t'}$$

$$\frac{t \;\rhd_\beta\; t'}{\pi_1 t \;\rhd_\beta\; \pi_1 t'} \qquad \frac{t \;\rhd_\beta\; t'}{\pi_2 t \;\rhd_\beta\; \pi_2 t'} \qquad \frac{t \;\rhd_\beta\; t'}{\langle t,u\rangle \;\rhd_\beta\; \langle t',u\rangle} \qquad \frac{u \;\rhd_\beta\; u'}{\langle t,u\rangle \;\rhd_\beta\; \langle t,u'\rangle}$$

$$\frac{t \;\rhd_\beta\; t'}{\texttt{case}\ t\ \{\} \;\rhd_\beta\; \texttt{case}\ t'\ \{\}} \qquad \frac{t \;\rhd_\beta\; t'}{\texttt{in}_1\, t \;\rhd_\beta\; \texttt{in}_1\, t'} \qquad \frac{t \;\rhd_\beta\; t'}{\texttt{in}_2\, t \;\rhd_\beta\; \texttt{in}_2\, t'}$$

$$\frac{t \;\rhd_\beta\; t'}{\texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\} \;\rhd_\beta\; \texttt{case}\ t'\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\}}$$

$$\frac{u \;\rhd_\beta\; u'}{\texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\} \;\rhd_\beta\; \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u' \mid \texttt{in}_2\, x \mapsto v\}}$$

$$\frac{v \;\rhd_\beta\; v'}{\texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\} \;\rhd_\beta\; \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v'\}}$$

Figure 5: The Relation of Full (Strong) β-Reduction.

We consider **types** given by the grammar

$$T, U \quad ::= \quad \kappa \quad \mid \quad U \to T \quad \mid \quad T \times U \quad \mid \quad T + U \quad \mid \quad \texttt{unit} \quad \mid \quad \texttt{void}$$

where κ ranges over some (unspecified) set of **base types**. The **typing relation** $\mathcal{E} \vdash t : T$ is defined by the rules of Fig. 6.

Exercise 4.25 (Booleans)**.** *Let* $\texttt{bool} := \texttt{unit} + \texttt{unit}$ *and*

$$\begin{array}{lcccc} \mathsf{true} & := & \texttt{in}_1\, \langle\rangle & : & \texttt{bool} \\ \mathsf{false} & := & \texttt{in}_2\, \langle\rangle & : & \texttt{bool} \end{array}$$

Define λ-terms $\texttt{neg} : \texttt{bool} \to \texttt{bool}$ *and* $\texttt{and} : \texttt{bool} \times \texttt{bool} \to \texttt{bool}$ *implementing resp. Boolean negation and conjunction.*

In particular, all Boolean functions are definable by λ-terms. This extends to any finite "base type".

$$(\textsc{Var})\ \frac{}{\mathcal{E} \vdash x : T}\ ((x:T) \in \mathcal{E}) \qquad (\texttt{unit}\text{-I})\ \frac{}{\mathcal{E} \vdash \langle\rangle : \texttt{unit}} \qquad (\texttt{void}\text{-E})\ \frac{\mathcal{E} \vdash t : \texttt{void}}{\mathcal{E} \vdash \texttt{case}_\bot\ t\ \{\} : T}$$

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x.t : U \to T} \qquad (\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash tu : T}$$

$$(\times\text{-I})\ \frac{\mathcal{E} \vdash t_1 : T_1 \qquad \mathcal{E} \vdash t_2 : T_2}{\mathcal{E} \vdash \langle t_1, t_2\rangle : T_1 \times T_2} \qquad (\times_1\text{-E})\ \frac{\mathcal{E} \vdash t : T_1 \times T_2}{\mathcal{E} \vdash \pi_1 t : T_1} \qquad (\times_2\text{-E})\ \frac{\mathcal{E} \vdash t : T_1 \times T_2}{\mathcal{E} \vdash \pi_2 t : T_2}$$

$$(+_1\text{-I})\ \frac{\mathcal{E} \vdash t : T_1}{\mathcal{E} \vdash \texttt{in}_1\, t : T_1 + T_2} \qquad (+_2\text{-I})\ \frac{\mathcal{E} \vdash t : T_2}{\mathcal{E} \vdash \texttt{in}_2\, t : T_1 + T_2}$$

$$(+\text{-E})\ \frac{\mathcal{E} \vdash t : T_1 + T_2 \qquad \mathcal{E}, x_1 : T_1 \vdash u_1 : U \qquad \mathcal{E}, x_2 : T_2 \vdash u_2 : U}{\mathcal{E} \vdash \texttt{case}\ t\ \{\texttt{in}_1\, x_1 \mapsto u_1 \mid \texttt{in}_2\, x_2 \mapsto u_2\} : U}$$

Figure 6: Typing Rules with Sums and Products.

Exercise 4.26 (Finite Base Types)**.** *We define the type* $\sum_{i=1}^n T_i$ *by induction on* $n \in \mathbb{N}$ *as*

$$\begin{array}{rcl} \sum_{i=1}^0 T_i & := & \texttt{void} \\ \sum_{i=1}^{n+1} T_i & := & T_1 + \sum_{i=1}^n T_{i+1} \end{array}$$

Given a finite set $A = \{a_1, \dots, a_n\}$*, the type* $\texttt{A}$ *is* $\sum_{i=1}^n \texttt{unit}$*. For each* $k = 1, \dots, n$*, let*

$$\texttt{a}_k \quad := \quad \underbrace{\texttt{in}_2 \cdots \texttt{in}_2}_{k-1\ \textit{times}}\, \texttt{in}_1\, \langle\rangle \quad : \quad \texttt{A}$$

Assume $A = \{a_1, \dots, a_n\}$ *and* $B = \{b_1, \dots, b_m\}$*. Show that for each function* $f : A \to B$ *there is a* λ*-term* $\texttt{f} : \texttt{A} \to \texttt{B}$ *such that*

$$\texttt{f}(\texttt{a}_k) \quad \rhd_\beta^* \quad \texttt{b}_\ell \qquad \textit{if and only if} \qquad f(a_k) \quad = \quad b_\ell$$

4.4.4. Main Properties

We now discuss the extension of §4.2.2 to product and sum types.

The following are proved exactly as their counterparts in §4.2.2.

Fact 4.27 (Inversion)**.**

(1) If $\mathcal{E} \vdash x : T$*, then* $(x : T) \in \mathcal{E}$*.*

(2) If $\mathcal{E} \vdash \lambda x.t : T$*, then* T *is of the form* $V \to U$ *and* $\mathcal{E}, x : V \vdash t : U$*.*

(3) If $\mathcal{E} \vdash tu : T$ *then there is some* U *such that* $\mathcal{E} \vdash t : U \to T$ *and* $\mathcal{E} \vdash u : U$*.*

(4) If $\mathcal{E} \vdash \langle t_1, t_2\rangle : T$ *then* T *is of the form* $U_1 \times U_2$ *and* $\mathcal{E} \vdash t_i : U_i$ *for each* $i = 1, 2$*.*

(5) If $\mathcal{E} \vdash \pi_i t : T_i$ then there is some T_{3-i} such that $\mathcal{E} \vdash t : T_1 \times T_2$.

(6) If $\mathcal{E} \vdash \langle\rangle : T$ then $T = \mathtt{unit}$.

(7) If $\mathcal{E} \vdash \mathtt{in}_i\, t : T$ then T is of the form $U_1 + U_2$ and $\mathcal{E} \vdash t : U_i$.

(8) If $\mathcal{E} \vdash \mathtt{case}\ t\ \{\mathtt{in}_1\, x_1 \mapsto u_1 \mid \mathtt{in}_2\, x_2 \mapsto u_2\} : T$, then there are some U_1, U_2 such that $\mathcal{E} \vdash t : U_1 + U_2$ and $\mathcal{E}, x_i : U_i \vdash u_i : T$ for $i = 1, 2$.

(9) If $\mathcal{E} \vdash \mathtt{case}_\perp\ t\ \{\} : T$ then $\mathcal{E} \vdash t : \mathtt{void}$.

Lemma 4.28 (Structural Properties)**.**

(Weakening) *If $\mathcal{E} \vdash t : T$ and $x \notin \mathrm{dom}(\mathcal{E})$ then $\mathcal{E}, x : U \vdash t : T$.*

(Contraction) *If $\mathcal{E}, x : U, y : U \vdash t : T$ then $\mathcal{E}, x : U \vdash t[x/y] : T$.*

(Exchange) *If $\mathcal{E}, x : U, \mathcal{E}', y : V, \mathcal{E}'' \vdash t : T$, then $\mathcal{E}, y : V, \mathcal{E}', x : U, \mathcal{E}'' \vdash t : T$.*

Lemma 4.29 (Substitution)**.** *If $\mathcal{E}, x : U \vdash t : T$ and $\mathcal{E} \vdash u : U$ then $\mathcal{E} \vdash t[u/x] : T$.*

Proposition 4.30 (Subject Reduction)**.** *If $\mathcal{E} \vdash t : T$ and $t \rhd_\beta u$ then $\mathcal{E} \vdash u : T$.*

We now turn to normalization. Theorem 4.12 (§4.2.2) extends to sums and products.

Theorem 4.31 (Strong Normalization)**.** *If $\mathcal{E} \vdash t : T$ then t is strongly β-normalizing (in the sense of Def. 3.33, §3.4.3).*

Theorem 4.31 is admitted until §5 (Thm. 5.32, §5.5). Let us now discuss some of its corollaries.

First, concerning confluence, in contrast with the pure λ-calculus we shall not be concerned with the case of the **untyped** extended λ-calculus. We shall thus content ourselves with confluence on **typed terms**, which can be obtained using Newman's Lemma 3.35 (§3.4.3) from Theorem 4.31 and local confluence (see [SU06, §4.5]). In the following, the notation $|\mathcal{E} \vdash T|$ refers to the obvious adaptation of Rem. 4.11 (§4.2.2).

Theorem 4.32. *For each typing context $\mathcal{E}$ and each type T, the relation $\rhd_\beta$ is confluent on $|\mathcal{E} \vdash T|$.*

Finally, we can state and prove the (relevant) analogue of Thm. 2.9 (§2.1). This relies on the following crucial fact on the shape of (typed) normal forms.

Lemma 4.33 (Closed Typed Normal Forms)**.** *If t is a closed typable λ-term in β-normal form, then t is of one of the following forms:*

$$\lambda x.u \qquad \langle\rangle \qquad \langle u, v\rangle \qquad \mathtt{in}_1 u \qquad \mathtt{in}_2 u$$

Proof. By induction on t. If t is of one of the above form, then we are done. Moreover, t cannot be a variable since it is closed. We inspect the other cases (implicitly relying on Fact 4.27).

Case of $t = uv$.

Then u would be a normal form since the relation $\rhd_\beta$ is closed under the rule

$$\frac{u \;\rhd_\beta\; u'}{u\,v \;\rhd_\beta\; u'\,v}$$

Since u is closed and typed, by induction hypothesis u would be of the form $\lambda x.w$. But this is impossible since $t = (\lambda x.w)v$ is assumed to be a normal form.

Cases of $t = \pi_i u$ ($i = 1, 2$).

Then u would be a normal form since the relation $\rhd_\beta$ is closed under the rule

$$\frac{u \;\rhd_\beta\; u'}{\pi_i u \;\rhd_\beta\; \pi_i u'}$$

Since u is closed and typed, by induction hypothesis u would be of the form $\langle v, w\rangle$. But this is impossible since $t = \pi_i\langle v, w\rangle$ is assumed to be a normal form.

Case of $t = \texttt{case}\ u\ \{\texttt{in}_1\, x \mapsto v \mid \texttt{in}_2\, x \mapsto w\}$.

Then u would be a normal form since the relation $\rhd_\beta$ is closed under the rule

$$\frac{u \;\rhd_\beta\; u'}{\texttt{case}\ u\ \{\texttt{in}_1\, x \mapsto v \mid \texttt{in}_2\, x \mapsto w\} \;\rhd_\beta\; \texttt{case}\ u'\ \{\texttt{in}_1 \mapsto v \mid \texttt{in}_2\, x \mapsto w\}}$$

Since u is closed and typed, by induction hypothesis u would be of the form $\texttt{in}_i\, v$. But this is impossible since t is assumed to be a normal form.

Case of $t = \texttt{case}_\bot\ u\ \{\}$.

Then u would be a normal form since the relation $\rhd_\beta$ is closed under the rule

$$\frac{u \;\rhd_\beta\; u'}{\texttt{case}_\bot\ u\ \{\} \;\rhd_\beta\; \texttt{case}_\bot\ u'\ \{\}}$$

Since u is closed, by induction hypothesis it must be of one of the form prescribed in the statement of the Lemma. But this is impossible since no term of these form can have type `void`. □

Combining Lem. 4.33 with Strong Normalization (Thm. 4.31) and Subject Reduction (Prop. 4.30) readily gives the following.

Corollary 4.34. *In the simply-typed λ-calculus with sums and products,*

(1) there is no λ-term t such that $\vdash t : \texttt{void}$;

(2) if $\vdash t : T_1 + T_2$ then there are some $i \in \{1, 2\}$ and some λ-term u_i such that $t \rhd^*_\beta \texttt{in}_i\, u_i$ and $\vdash u_i : T_i$.

Corollary 4.34 is proven without relying on Strong Normalization (Thm. 4.31) in §5.3 (Cor. 5.11).

	Types	$\equiv$	Formulae of Intuitionistic Prop. Logic	
(base types)	κ	$\equiv$	p	(atomic propositions)
(functions)	$(-) \to (-)$	$\equiv$	$(-) \Rightarrow (-)$	(implications)
(products)	$(-) \times (-)$	$\equiv$	$(-) \wedge (-)$	(conjunctions)
(unit)	`unit`	$\equiv$	$\top$	(truth)
(sums)	$(-) + (-)$	$\equiv$	$(-) \vee (-)$	(int. disjunctions)
(void)	`void`	$\equiv$	$\bot$	(falsity)

Table 2: The Curry-Howard Correspondence: Types and Formulae.

4.4.5. Curry-Howard Correspondence

We now proceed to the Curry-Howard correspondence for the full intuitionistic propositional logic NJ_0.

We begin with the identification of **types** with **formulae**. First, recall their respective grammars:

$$
\begin{array}{lclcccccccccc}
T, U & ::= & \kappa & \mid & U \to T & \mid & T \times U & \mid & T + U & \mid & \texttt{unit} & \mid & \texttt{void} \\
A, B & ::= & \mathsf{p} & \mid & B \Rightarrow A & \mid & A \wedge B & \mid & A \vee B & \mid & \top & \mid & \bot
\end{array}
$$

Extending §4.3, we shall further identify:

- conjunctions $(-) \wedge (-)$ with product types $(-) \times (-)$ (and $\top$ with `unit`);
- disjunctions $(-) \vee (-)$ with the sum types $(-) + (-)$ (and $\bot$ with `void`).

The resulting correspondence between types and formulae is summarized in Table 2. We may thus consider that **types are formulae of propositional logic**.

Notation 4.35 (Types as Formulae)**.** *Extending Notation 4.14 (§4.3), for the remaining of this §4.4.5 we assume that* ***types*** *given by the grammar of formulae:*

$$
A, B \quad ::= \quad \mathsf{p} \quad \mid \quad A \Rightarrow B \quad \mid \quad A \wedge B \quad \mid \quad A \vee B \quad \mid \quad \top \quad \mid \quad \bot
$$

As depicted in Fig. 7, erasing λ-terms from typing rules exactly gives the derivation rules of NJ_0. It follows that **typed λ-terms** can be identified with **proofs**:

Intuitionistic Logic (NJ_0)		Simply-Typed λ-Calculus (with $+$, $\times$)
Formulae	$\equiv$	Types
Proofs	$\equiv$	Typed λ-Terms

Remark 4.36 (Proof-Terms)**.** *Under the Curry-Howard correspondence, typed λ-terms are often called* ***proof-terms****.*

In particular, we have the following obvious extension of Thm. 4.15 (§4.3).

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i} \quad \rightsquigarrow \quad \frac{}{A_1, \ldots, A_n \vdash A_i} \qquad (\text{Ax})$$

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash \langle\rangle : \top} \quad \rightsquigarrow \quad \frac{}{A_1, \ldots, A_n \vdash \top} \qquad (\top\text{-I})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : \bot}{x_1 : A_1, \ldots, x_n : A_n \vdash \texttt{case}_\bot\ t\ \{\} : B} \quad \rightsquigarrow \quad \frac{A_1, \ldots, A_n \vdash \bot}{A_1, \ldots, A_n \vdash B} \qquad (\bot\text{-E})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n, x : A \vdash t : B}{x_1 : A_1, \ldots, x_n : A_n \vdash \lambda x.t : A \Rightarrow B} \quad \rightsquigarrow \quad \frac{A_1, \ldots, A_n, A \vdash B}{A_1, \ldots, A_n \vdash A \Rightarrow B} \qquad (\Rightarrow\text{-I})$$

$$\frac{\begin{array}{c} x_1 : A_1, \ldots, x_n : A_n \ \vdash\ t : A \Rightarrow B \\ x_1 : A_1, \ldots, x_n : A_n \ \vdash\ u : A \end{array}}{x_1 : A_1, \ldots, x_n : A_n \vdash tu : B} \quad \rightsquigarrow \quad \frac{\begin{array}{c} A_1, \ldots, A_n \ \vdash\ A \Rightarrow B \\ A_1, \ldots, A_n \ \vdash\ A \end{array}}{A_1, \ldots, A_n \vdash B} \qquad (\Rightarrow\text{-E})$$

$$\frac{\begin{array}{c} x_1 : A_1, \ldots, x_n : A_n \ \vdash\ t_1 : B_1 \\ x_1 : A_1, \ldots, x_n : A_n \ \vdash\ t_2 : B_2 \end{array}}{x_1 : A_1, \ldots, x_n : A_n \vdash \langle t_1, t_2\rangle : B_1 \wedge B_2} \quad \rightsquigarrow \quad \frac{\begin{array}{c} A_1, \ldots, A_n \ \vdash\ B_1 \\ A_1, \ldots, A_n \ \vdash\ B_2 \end{array}}{A_1, \ldots, A_n \vdash B_1 \wedge B_2} \qquad (\wedge\text{-I})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B_1 \wedge B_2}{x_1 : A_1, \ldots, x_n : A_n \vdash \pi_i t : B_i} \quad \rightsquigarrow \quad \frac{A_1, \ldots, A_n \vdash B_1 \wedge B_2}{A_1, \ldots, A_n \vdash B_i} \qquad (\wedge_i\text{-E})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B_i}{x_1 : A_1, \ldots, x_n : A_n \vdash \texttt{in}_i\, t : B_1 \vee B_2} \quad \rightsquigarrow \quad \frac{A_1, \ldots, A_n \vdash B_i}{A_1, \ldots, A_n \vdash B_1 \vee B_2} \qquad (\vee_i\text{-I})$$

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash t : B_1 \vee B_2 \qquad \begin{array}{c} x_1 : A_1, \ldots, x_n : A_n,\ x : B_1 \ \vdash\ u_1 : C \\ x_1 : A_1, \ldots, x_n : A_n,\ x : B_2 \ \vdash\ u_2 : C \end{array}}{x_1 : A_1, \ldots, x_n : A_n \vdash \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u_1 \mid \texttt{in}_2\, x \mapsto u_2\} : C} \qquad (\vee\text{-E})$$

$$\rightsquigarrow \qquad \frac{A_1, \ldots, A_n \vdash B_1 \vee B_2 \qquad \begin{array}{c} A_1, \ldots, A_n,\ B_1 \ \vdash\ C \\ A_1, \ldots, A_n,\ B_2 \ \vdash\ C \end{array}}{A_1, \ldots, A_n \vdash C}$$

Figure 7: The Curry-Howard Correspondence: Typing and Deduction Rules.

Theorem 4.37 (Curry-Howard Correspondence)**.** *The sequent $A_1, \dots, A_n \vdash A$ is derivable in NJ_0 if and only if there is a λ-term t such that $x_1 : A_1, \dots, x_n : A_n \vdash t : A$.*

Example 4.38. *We have $\vdash t : (A \vee \neg A) \Rightarrow (\neg A \Rightarrow A) \Rightarrow A$ where*

$$\begin{array}{rcl} t & : & (A \vee \neg A) \Rightarrow (\neg A \Rightarrow A) \Rightarrow A \\ & := & \lambda e.\ \lambda c.\ \texttt{case}\ e\ \{ \\ & & \quad \texttt{in}_1\, x_1 \mapsto x_1 \\ & & \mid\ \texttt{in}_2\, x_2 \mapsto c\, x_2\ \} \end{array}$$

since

$$\begin{array}{rcl} e : A \vee \neg A,\ c : \neg A \Rightarrow A & \vdash & \texttt{case}\ e\ \{ \\ & & \quad \texttt{in}_1\, x_1 \mapsto x_1 \\ & & \mid\ \texttt{in}_2\, x_2 \mapsto c\, x_2\ \} \quad : \quad A \end{array}$$

Together with Cor. 4.34 (§4.4.4), Thm. 4.37 gives the remaining part of Thm. 2.9 (§2.1).

Corollary 4.39. *In intuitionistic propositional logic (NJ_0),*

(1) $\vdash \bot$ is not derivable;

(2) if $\vdash A \vee B$ then either $\vdash A$ or $\vdash B$.

Similarly as in §4.3, we consider some examples from §2.

Exercise 4.40. *In each case below, give a λ-term t of the prescribed type:*

(1) $\vdash t : ((\neg A \Rightarrow A) \Rightarrow A) \Rightarrow \neg\neg A \Rightarrow \neg A$

(2) $\vdash t : (A \vee \neg A) \Rightarrow \neg\neg A \Rightarrow A$

(3) $\vdash t : \neg\neg(A \vee \neg A)$

Exercise 4.41. *In each case below, give a λ-term t of the prescribed type:*

(1) $\vdash t : B \wedge (A_1 \vee A_2) \Rightarrow (B \wedge A_1) \vee (B \wedge A_2)$

(2) $\vdash t : (B \wedge A_1) \vee (B \wedge A_2) \Rightarrow B \wedge (A_1 \vee A_2)$

Remark 4.42. *The proof terms of Ex. 4.41 enforce a form of distributive law. Their mere existence is already due to deep semantic reasons, related to the interaction of $\Rightarrow/\rightarrow$ with $\wedge/\times$ and $\vee/+$.*

Let us say a few words on this, in a simple setting which actually forms the basis the of algebraic semantics of intuitionistic propositional logic (see e.g. [SU06, §2.4]). The following is somewhat beyond the scope of this course.

Define the preorder $\leq$ on formulae as $A \leq B$ if $A \vdash B$ in NJ_0. The relation $\leq$ becomes a partial order on formulae quotiented by the equivalence relation

$$A \equiv B \quad \textit{iff} \quad A \leq B \textit{ and } B \leq A$$

The poset of quotiented formulae is actually a ***lattice****, i.e. it has all finite sups and infs (see [SU06, Def. 2.3.1]) given resp. by the action of* $\vee$ *(with unit* $\bot$*) and* $\wedge$ *(with unit* $\top$*) on quotiented formulae.*

Now, Ex. 4.41 says that this lattice is ***distributive*** *([SU06, Def. 2.3.5]) in the sense that it satisfies the following law:*[9]

$$(C \vee B) \wedge A \quad = \quad (C \wedge A) \vee (B \wedge A)$$

(we confuse formulae with their equivalence classes). But beware that not every lattice is distributive! A well-known counter example is the lattice M_3*, which can be depicted as follows:*

$$\begin{tikzcd}
 & \top \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
a \arrow[dr, dash] & b \arrow[d, dash] & c \arrow[dl, dash] \\
 & \bot &
\end{tikzcd}$$

The distributive law above may be read as the preservation of $(-) \vee (-)$ *by* $(-) \wedge A$*, and actually follows from the following property of logical implication* $(\Rightarrow)$*:*[10]

$$B \wedge A \;\leq\; C \qquad \textit{iff} \qquad B \;\leq\; A \Rightarrow C$$

Remark 4.43. *Alternative syntaxes consider* `case`*'s of the following form (e.g. [Gal95]):*

$$\frac{\mathcal{E} \vdash t : A_1 + A_2 \qquad \mathcal{E}, x_1 : A_1 \vdash u_1 : B \qquad \mathcal{E}, x_2 : A_2 \vdash u_2 : B}{\mathcal{E} \vdash \mathtt{case}\ t\ \{\mathtt{in}_1 \mapsto \lambda x_1.u_1 \mid \mathtt{in}_2 \mapsto \lambda x_2.u_2\} : B}$$

Remark 4.44 (On the BHK Interpretation)**.** *The Curry-Howard correspondence can be seen as an instance of the BHK interpretation (see §4.1). More precisely, under the Curry-Howard correspondence, Thm. 4.31 and Lem. 4.33 (§4.4.4) give the following (where by "*proof*" we understand "*proof term in β-normal form*"):*

- *a "*proof*" of* $A_1 \wedge A_2$ *is a pair of a "*proof*" of* A_1 *and a "*proof*" of* A_2*;*
- *a "*proof*" of* $A_1 \vee A_2$ *contains an* $i \in \{1, 2\}$ *and a "*proof*" of* A_i*;*
- *a "*proof*" of* $A \Rightarrow B$ *is a "function" taking a "*proof*" of* A *to a "*proof*" of* B*;*
- *there is no "*proof*" of* $\bot$*.*

Above, the word "function" can be understood to refer to those computable functions f *such that for some* λ*-term* $\vdash \lambda x.t : A \Rightarrow B$*,* f *takes a "*proof*"* $\vdash u : A$ *to the "*proof*" of* B *obtained by* β*-normalizing* $\vdash t[u/x] : B$*.*

[9] The two laws mentioned in [SU06, Def. 2.3.5] are actually equivalent in any lattice.

[10] See *e.g.* [Awo10, §6.3] and [LS86, Ex. I.7.4 & §I.8] (outside the scope of this course).

Remark 4.45 (Extension to Classical Logic)**.** *With further enrichment of the simply-typed λ-calculus, the Curry-Howard correspondence actually extends to* ***classical*** *propositional logic, see [SU06, §6] (also [AC98, §8.5]).* ***Beware*** *that the resulting correspondence* ***does not*** *extend Tab. 2. In particular, interpreting (classical) disjunctions as sum types requires to switch to a* ***call-by-value*** *λ-calculus.*[11]

5. Normalization for Simple Types

5.1. Introduction

In the case of the simply typed λ-calculus, most normalization properties are provable by direct induction on typed terms (see *e.g.* [SU06, §3.5]). However, these proofs are often difficult (and sometime impossible) to generalize.

We are going to discuss proofs using a method called **reducibility**. Reducibility applies to (much) stronger type systems than simple types; it is actually the only known way to prove the termination of any type system extending Girard's **System F**[12] §9. Besides [SU06, §11], see [GLT89, Bar92, Kri93], as well as [Pie02, §23].

The reducibility method follows a general pattern issued from **realizability**. Realizability in particular gives (classes of) models of higher-order arithmetic and set theory, in intuitionistic settings as well as in classical ones. It is even possible to validate axioms compatible with intuitionistic logic but which are classically false (§8.3)! See [TvD88a, Chap. 4, §4] and also [Kri09, Kri01].

Another version of realizability, called **logical relations**, is a widespread tool to prove various properties of (models of) higher-order functional programming languages.[13]

We present the basic mechanisms of the technique, and illustrate it with proofs of normalization results of increasing strengths. We introduce the ingredients steps by steps. After some notational preliminaries (§5.2), we begin in §5.3 with a proof of the Disjunction Property (in the form of Cor. 4.34, §4.4.4). In §5.4, we strengthen this argument to a proof of Weak Normalization for the Weak Head Reduction (to be defined in §5.3.1). We finally present in §5.5 the adaptations needed in order to obtain a proof of Strong Normalization for the Full Strong Reduction.

Standard references include [GLT89, Bar92, Kri93].

5.2. Notational Preliminaries

The reference λ-calculus for this §5 is the simply-typed λ-calculus with sums and products described in §4.4.3. Extending §3.2.1 (actually Notation 3.7, §3.2.4) we write Λ for the set of extended λ-terms of §4.4.3 (modulo α-conversion). Recall that $\mathcal{X}$ is the set of variables for λ-terms.

We generalize the operation of capture-avoiding substitution (Def. 3.8, §3.3.1) to the following notion of **simultaneous** capture-avoiding substitution.

[11] See [LS86, §I.8] and [Sel01] (outside the scope of this course).
[12] A.k.a. Reynolds' **Polymorphic λ-Calculus**.
[13] See *e.g.* [AC98, §4.5] (outside the scope of this course).

$$
\begin{array}{rcll}
x\sigma & := & \sigma(x) & \text{if } x \in \mathrm{dom}(\sigma) \\
x\sigma & := & x & \text{if } x \notin \mathrm{dom}(\sigma) \\
(\lambda x.t)\sigma & := & \lambda x.t\sigma & \text{if } x \notin \mathrm{dom}(\sigma) \cup \mathrm{FV}(\mathrm{img}(\sigma)) \\
(tu)\sigma & := & t\sigma\, u\sigma & \\
\langle\rangle\sigma & := & \langle\rangle & \\
\langle t, u\rangle\sigma & := & \langle t\sigma, u\sigma\rangle & \\
(\pi_i t)\sigma & := & \pi_i t\sigma & \text{for } i = 1, 2 \\
(\mathtt{in}_i\, t)\sigma & := & \mathtt{in}_i\, t\sigma & \text{for } i = 1, 2 \\
\multicolumn{4}{l}{\text{if } x_1, x_2 \notin \mathrm{dom}(\sigma) \cup \mathrm{FV}(\mathrm{img}(\sigma)),} \\
\big(\mathtt{case}\ t\ \{\mathtt{in}_1\, x_1 \mapsto u_1 \mid \mathtt{in}_2\, x_2 \mapsto u_2\}\big)\sigma & := & \mathtt{case}\ t\sigma\ \{\mathtt{in}_1\, x_1 \mapsto u_1\sigma \mid \mathtt{in}_2\, x_2 \mapsto u_2\sigma\} & \\
\big(\mathtt{case}_\perp\ t\ \{\}\big)\sigma & := & \mathtt{case}_\perp\ t\sigma\ \{\} &
\end{array}
$$

Figure 8: Simultaneous Capture-Avoiding Substitution.

Definition 5.1 (Simultaneous Substitution)**.** *Let σ be a* partial *function $\mathcal{X} \rightharpoonup \Lambda$. The **simultaneous capture-avoiding substitution** $t\sigma$ is defined by induction on t in Fig. 8, where $\mathrm{FV}(\mathrm{img}(\sigma))$ stands for $\bigcup_{x\in\mathrm{dom}(\sigma)} \mathrm{FV}(\sigma(x))$.*

Notation 5.2.

*(1) A $\sigma : \mathcal{X} \rightharpoonup \Lambda$ as in Def. 5.1 is often called a **substitution**.*

(2) When $x \notin \mathrm{dom}(\sigma)$, we write $\sigma[u/x]$ for the substitution taking x to u and equal to σ everywhere else.

Note that $t(\sigma[u/x]) = (t\sigma)[u/x]$ if $x \notin \mathrm{FV}(\mathrm{img}(\sigma))$; we can thus unambiguously write $t\sigma[u/x]$ in this case.

(3) When $\mathrm{dom}(\sigma) = \{x_1, \dots, x_n\}$ we often write $[\sigma(x_1)/x_1, \dots, \sigma(x_n)/x_n]$ for σ, and thus $t[\sigma(x_1)/x_1, \dots, \sigma(x_n)/x_n]$ for $t\sigma$.

In particular, $[u_1/x_1, \dots, u_n/x_n]$ stands for the substitution of domain $\{x_1, \dots, x_n\}$ which takes x_i to u_i.

Notation 5.3 (Types)**.** *For this §5, we write Ty for the set of simple types with sums and products (§4.4.3), with base types κ ranging over a determined set $\mathcal{K}$.*

Finally, we use the notions related to normalization introduced in Def. 3.33 (§3.4.3).

5.3. A First Normalization Result: The Disjunction Property

In this §5.3, we present the basic mechanism of the reducibility technique, and illustrate it with a proof of the Disjunction Property, in the form of Cor. 4.34 (§4.4.4). See Cor. 5.11 (§5.3.3) below.

5.3.1. Weak Head Reduction

We begin by noting that in order to get Lem. 4.33 (4.4.4) on Closed Typed Normal Forms, we actually do not need the full strong reduction relation $\rhd_\beta$ but only the weaker (and simpler) **weak head reduction**. This shall greatly simplify the proof of the disjunction property.

Recall the **basic** reduction relation $\rhd_0$ from Fig. 5 (§4.4.3). If we inspect the proof of Lem. 4.33, we remark that beside $\rhd_0 \subseteq \rhd_\beta$, we only use the closure of $\rhd_\beta$ under the following rules:

$$\frac{u \ \rhd_\beta \ u'}{u \ v \ \rhd_\beta \ u' \ v} \qquad \frac{u \ \rhd_\beta \ u'}{\pi_i \ u \ \rhd_\beta \ \pi_i \ u'} \qquad \frac{u \ \rhd_\beta \ u'}{\texttt{case}_\bot \ u \ \{\} \ \rhd_\beta \ \texttt{case}_\bot \ u' \ \{\}}$$

$$\frac{u \ \rhd_\beta \ u'}{\texttt{case} \ u \ \{\texttt{in}_1 \, x \mapsto v \mid \texttt{in}_2 \, x \mapsto w\} \ \rhd_\beta \ \texttt{case} \ u' \ \{\texttt{in}_1 \, x \mapsto v \mid \texttt{in}_2 \, x \mapsto w\}}$$

The least relation containing $\rhd_0$ and closed under the above rules is the **weak head reduction**. For the following, it will be convenient to define it by "plugging" the basic reduction $\rhd_0$ into some term-contexts of a very constrained form, ensuring that reduction steps can only occur in weak head position. These contexts are called **elimination contexts**. They are the terms with a hole (noted $[\,]$) inductively defined as follows:

$$\begin{array}{rcl} E[\,] \in \mathrm{Elim} & ::= & [\,] \quad | \quad E[\,]\ t \quad | \quad \pi_1 \ E[\,] \quad | \quad \pi_2 \ E[\,] \\ & | & \texttt{case} \ E[\,] \ \{\texttt{in}_1 \, x \mapsto u \mid \texttt{in}_2 \, x \mapsto v\} \quad | \quad \texttt{case} \ E[\,] \ \{\} \end{array}$$

We write $E[t]$ for the term obtained by substituting t for the (unique occurrence of) the hole $[\,]$ in $E[\,]$. Note that elimination contexts are stable by composition: if $E[\,], F[\,] \in \mathrm{Elim}$, then $E[F[\,]] \in \mathrm{Elim}$.

Definition 5.4 (Weak Head Reduction)**.** *The relation $\rhd_{\mathrm{wh}}$ of **weak head reduction** is defined as $t \rhd_{\mathrm{wh}} u$ if and only if there is an elimination context $E[\,]$ and terms v, w such that $t = E[v]$, $u = E[w]$ and $v \rhd_0 w$.*

A direct consequence of the definition is that if $t \rhd_{\mathrm{wh}} u$, then we also have $E[t] \rhd_{\mathrm{wh}} E[u]$ for all elimination context $E[\,]$. This leads to a analogue of Lem. 4.33 for the closed normal forms of $\rhd_{\mathrm{wh}}$. A **weak head normal form** is a normal form for $\rhd_{\mathrm{wh}}$.

Lemma 5.5 (Closed Normal Forms)**.** *If t is a closed typable term in weak head normal form, then t is of one of the following forms:*

$$\lambda x.u \qquad \langle\rangle \qquad \langle u, v\rangle \qquad \texttt{in}_1 \, u \qquad \texttt{in}_2 \, u$$

Proof. Exercise! □

Remark 5.6. *The weak head reduction a "call-by-name" reduction strategy since arguments of β-redexes are only evaluated when they appear in head position, and never before contracting the β-redex. This is also a strategy for **weak** reduction: no redex is contracted under an abstraction. Such reduction strategies are very useful in practice, since they can be easily implemented with abstract machines.*

5.3.2. Type Interpretations and Adequacy

The general idea is the following: Assume that we want to prove that all typable terms $\vdash t : T$ satisfy a given property P. This property may be difficult to prove by a direct induction on the term structure or on the typing derivations. This is typically the case of termination.

Now, think of P as a subset of Λ, *i.e.* $P \subseteq \Lambda$. The idea to find a map $[\![-]\!]$, mapping each type $T \in \mathrm{Ty}$ to a set of terms satisfying P: $[\![T]\!] \subseteq P$. If we achieve to show that $\vdash t : T$ implies $t \in [\![T]\!]$, then we are done. We call $[\![-]\!] : \mathrm{Ty} \to \mathcal{P}(\Lambda)$ a **type interpretation**. If $\vdash t : T$ implies $t \in [\![T]\!]$, then $[\![-]\!]$ will be called adequate.

Definition 5.7 (Adequacy)**.** *Let* $[\![-]\!] : \mathrm{Ty} \to \mathcal{P}(\Lambda)$.

(1) Given a substitution σ and a typing context $\mathcal{E}$, we write $\sigma \models_{[\![-]\!]} \mathcal{E}$ if $\mathrm{dom}(\mathcal{E}) \subseteq \mathrm{dom}(\sigma)$ *and $\sigma(x) \in [\![T]\!]$ for all $(x : T) \in \mathcal{E}$.*

*(2) A type interpretation $[\![-]\!]$ is **adequate** if $t\sigma \in [\![T]\!]$ whenever $\mathcal{E} \vdash t : T$ and $\sigma \models_{[\![-]\!]} \mathcal{E}$.*

Let us now discuss some general conditions to get an adequate type interpretation. In most (if not all) cases, type interpretations $[\![-]\!] : \mathrm{Ty} \to \mathcal{P}(\Lambda)$ are defined by induction on the type structure, and adequacy is proved by induction on typing derivations.

Without making any assumption on $[\![-]\!]$, we can already look at what happens in the base case of an adequacy proof. Consider thus the case of the rule

$$(\textsc{Var})\ \frac{(x : T) \in \mathcal{E}}{\mathcal{E} \vdash x : T}$$

But if $\sigma \models_{[\![-]\!]} \mathcal{E}$ then by definition we have $\sigma(x) \in [\![T]\!]$. Hence, **any** type interpretation validates the $(\textsc{Var})$ rule.

All other typing rules involve a type constructor. Let us be a bit more precise about $[\![-]\!]$. Assume that each base type $\kappa \in \mathcal{K}$ has a given interpretation $[\![\kappa]\!] \in \mathcal{P}(\Lambda)$. Assume moreover that we have attributed an interpretation $[\![\mathtt{unit}]\!]$ and $[\![\mathtt{void}]\!]$ to the unit types `unit` and `void`. The interpretation of an arbitrary type is then inductively defined using an operator on $\mathcal{P}(\Lambda)$ per connective. The following operators $\mathrel{\square\!\!\rightarrow}$ and $\boxtimes$ correspond to the type constructors $(-) \to (-)$ and $(-) \times (-)$.

Definition 5.8. *Given $A, B \in \mathcal{P}(\Lambda)$, let*

$$\begin{array}{rcl} A \mathrel{\square\!\!\rightarrow} B & := & \{t \mid (\forall u \in A)(tu \in B)\} \\ A \boxtimes B & := & \{t \mid \pi_1 t \in A \quad \textit{and} \quad \pi_2 t \in B\} \end{array}$$

Assume now that $[\![-]\!]$ is such that for any types T, U, we have:

$$\begin{array}{rcl} [\![U \to T]\!] & = & [\![U]\!] \mathrel{\square\!\!\rightarrow} [\![T]\!] \\ [\![T \times U]\!] & = & [\![T]\!] \boxtimes [\![U]\!] \end{array}$$

It is not difficult to see that such an interpretation validates the elimination rules of $\to$ and $\times$. Consider for instance the case of

$$(\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash tu : T}$$

Let $\sigma \models_{[\![-]\!]} \mathcal{E}$. By induction hypothesis $t\sigma \in [\![U]\!] \mathrel{\Box\!\!\rightarrow} [\![T]\!]$ and $u\sigma \in [\![U]\!]$, and by definition of $\mathrel{\Box\!\!\rightarrow}$ we get $(t\sigma)(u\sigma) \in [\![T]\!]$. Then we are done since $(tu)\sigma = (t\sigma)(u\sigma)$.

There is however a difficulty with the introduction rules. Consider for instance

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x.t : U \to T}$$

Let $\sigma \models_{[\![-]\!]} \mathcal{E}$. Note that we can assume $x \notin \mathrm{FV}(\mathrm{img}(\sigma))$. We thus have $(\lambda x.t)\sigma = \lambda x.(t\sigma)$. By definition of $\mathrel{\Box\!\!\rightarrow}$, we must show that $(\lambda x.(t\sigma))u \in [\![T]\!]$ whenever $u \in [\![U]\!]$. Let $u \in [\![U]\!]$. By induction hypothesis, we have $(t\sigma)[u/x] = t\sigma[u/x] \in [\![T]\!]$. Hence, in order to conclude, we would like to deduce $(t\sigma)[u/x] \in [\![T]\!]$ from $(\lambda x.t\sigma)u \in [\![T]\!]$.

A slight generalization would lead us to require that for all $T \in \mathrm{Ty}$, $[\![T]\!]$ satisfies the following invariant:

(stability by expansion) if $u \in [\![T]\!]$ and $t \rhd_0 u$ then $t \in [\![T]\!]$.

Since $[\![-]\!]$ is defined by induction on types, we would like this invariant to be preserved by $\mathrel{\Box\!\!\rightarrow}$ and $\boxtimes$. Consider for instance the case of $\mathrel{\Box\!\!\rightarrow}$, and let us look at what we need to impose on $A, B \in \mathcal{P}(\Lambda)$ in order to ensure the stability by expansion of $A \mathrel{\Box\!\!\rightarrow} B$. Let $u \in A \mathrel{\Box\!\!\rightarrow} B$ and $t \rhd_0 u$. We have to show that for all $v \in A$, we have $tv \in B$ under the assumption that $uv \in B$. In other word, B must be stable by expansion not only for $\rhd_0$, but also for a relation $\rhd$ containing $\rhd_0$ and closed under the context rule

$$\frac{t \ \rhd\ u}{t\ v \ \rhd\ u\ v}$$

By iterating the argument, we would (almost) arrive at the following: for all type T, we ask $[\![T]\!]$ to be **stable by weak head expansion**:

- $(\mathcal{WHE})$ if $u \in [\![T]\!]$ and $t \rhd_{\mathrm{wh}} u$ then $t \in [\![T]\!]$.

5.3.3. Proof of the Disjunction Property

In order to complete the definition of an adequate type interpretation, we have to devise interpretations for the unit types `unit` and `void`, as well as for the sum types constructor $(-) + (-)$. For the moment, assume that

$$[\![T + U]\!] = [\![T]\!] \oplus [\![U]\!] \qquad [\![\texttt{unit}]\!] = \mathbf{1} \qquad [\![\texttt{void}]\!] = \mathbf{0}$$

where $\oplus$, $\mathbf{1}$ and $\mathbf{0}$ are defined as:

$$\begin{array}{rcl} A \oplus B & := & \big\{t \mid (\exists u)\big(t \rhd^*_{\mathrm{wh}} (\texttt{in}_1\, u) \ \ \&\ \ u \in A\big) \quad \text{or} \quad (\exists u)\big(t \rhd^*_{\mathrm{wh}} (\texttt{in}_2\, u) \ \ \&\ \ u \in B\big)\big\} \\ \mathbf{1} & := & \{t \mid t \rhd^*_{\mathrm{wh}} \langle\rangle\} \\ \mathbf{0} & := & \emptyset \end{array}$$

These definitions follow a pattern different from what we have already seen: they are **by introduction** (or **positive**) rather than **by elimination** (or **negative**). We are thus in a somehow dual situation w.r.t. the cases of $\mathrel{\Box\!\!\rightarrow}$ and $\boxtimes$. In particular, it is easy to see that $\oplus$ and $\mathbf{1}$ directly validate the introduction rules of $(-) + (-)$ and `unit`, while we rely on $(\mathcal{WHE})$ for the elimination rule of $(-) + (-)$:

Proposition 5.9. *Let $A, B, C \in \mathcal{P}(\Lambda)$ such that C satisfies $(\mathcal{WHE})$. Let t, v, w such that $t \in A \oplus B$, $v[u/x] \in C$ for all $u \in A$ and $w[u/x] \in C$ for all $u \in B$. Then $\texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto v \mid \texttt{in}_2\, x \mapsto w\} \in C$.*

Proof. Exercise! □

Consider now the case of $\mathbf{0}$. It trivially satisfies $(\mathcal{WHE})$. Let us see that it also validates the elimination rule of `void`

$$(\texttt{void-E})\ \frac{\mathcal{E} \vdash t : \texttt{void}}{\mathcal{E} \vdash \texttt{case}\ t\ \{\} : T}$$

Let $\sigma \models_{[\![-]\!]} \mathcal{E}$. We have to show that $\texttt{case}\ (t\sigma)\ \{\} \in [\![T]\!]$, while we have no assumption on T! It can be for instance `void`, while $[\![\texttt{void}]\!] = \emptyset$. But the induction hypothesis tells us that $t\sigma \in [\![\texttt{void}]\!] = \emptyset$. This is a contradiction, from which follows anything; in particular $\texttt{case}\ (t\sigma)\ \{\} \in [\![T]\!]$ for any type T.

Putting everything together, we have the following:

Theorem 5.10 (Adequacy)**.** *Let $[\![-]\!]$ be a type interpretation such that*

(i) $[\![\kappa]\!]$ satisfies $(\mathcal{WHE})$ for all $\kappa \in \mathcal{K}$, and

(ii) $[\![\texttt{unit}]\!] = \mathbf{1}$ and $[\![\texttt{void}]\!] = \mathbf{0}$, and

(iii) for all types T, U,

$$[\![U \to T]\!] = [\![U]\!] \mathrel{\square\!\!\to} [\![T]\!] \qquad [\![T \times U]\!] = [\![T]\!] \boxtimes [\![U]\!] \qquad [\![T + U]\!] = [\![T]\!] \oplus [\![U]\!]$$

Then,

(1) for all type T, $[\![T]\!]$ satisfies $(\mathcal{WHE})$, and

(2) $[\![-]\!]$ is adequate.

Proof. Exercise! □

Let us now look at what can be said from Thm. 5.10. At first sight, it may seem that we have imposed very few properties on $[\![-]\!]$. It is nevertheless enough to obtain the Disjunction Property.

Corollary 5.11 (Cor. 4.34 (§4.4.4))**.** *In the simply-typed λ-calculus with sums and products,*

(1) there is no λ-term t such that $\vdash t : \texttt{void}$;

(2) if $\vdash t : T_1 + T_2$ then there are some $i \in \{1, 2\}$ and some λ-term u_i such that $t \rhd^*_\beta \texttt{in}_i\, u_i$ and $\vdash u_i : T_i$.

Proof. Exercise! □

We thus have proved the Disjunction Property. However, Thm. 5.10 gives **no** information on the normalization of typed terms of **arbitrary** types.

5.4. Reducibility Candidates

In this section, we adapt the basic material of §5.3 to prove (weak) normalization of typed terms of **arbitrary types**. To keep things simple, we consider the weak normalization of weak head reduction $\rhd_{\mathrm{wh}}$.

Warning 5.12. *In this §5.4, by **weak normalization** we mean weak normalization w.r.t. $\rhd_{\mathrm{wh}}$. Write $\mathcal{WN}$ for the set of weakly normalizable terms w.r.t. $\rhd_{\mathrm{wh}}$.*

In order to show that $\vdash t : T$ implies $t \in \mathcal{WN}$, we only have to find a type interpretation $[\![-]\!]$ such that $[\![T]\!] \subseteq \mathcal{WN}$ for all type T. First, note that requiring $[\![T]\!] \subseteq \mathcal{WN}$ does not conflict with $(\mathcal{WHE})$.

Lemma 5.13. *Let $A \subseteq \mathcal{WN}$. Then $\{t \mid \exists(u \in A)(t \rhd^*_{\mathrm{wh}} u)\}$ is the least subset of $\mathcal{WN}$ which contains A and satisfies $(\mathcal{WHE})$.*

Consider now a type interpretation $[\![-]\!]$, which satisfies the hypotheses of Thm. 5.10 and such that moreover $[\![\kappa]\!] \subseteq \mathcal{WN}$ for all κ. Note that $[\![\mathtt{unit}]\!], [\![\mathtt{void}]\!] \subseteq \mathcal{WN}$.

Let us check whether the requirement $[\![T]\!] \subseteq \mathcal{WN}$ is preserved by the (interpretations of the) type constructors. The case of $\oplus$ is trivial since we have $A \oplus B \subseteq \mathcal{WN}$ for all $A, B \in \mathcal{P}(\Lambda)$. Moreover, it is not difficult to see that $A \boxtimes B \subseteq \mathcal{WN}$ provided $A, B \subseteq \mathcal{WN}$.

Exercise 5.14. *Show that $A \boxtimes B \subseteq \mathcal{WN}$ whenever $A, B \subseteq \mathcal{WN}$. What property of $\rhd_{\mathrm{wh}}$ do you use?*

However, there is a problem with $\mathrel{\square\!\!\rightarrow}$ since

$$\mathbf{0} \mathrel{\square\!\!\rightarrow} A \quad = \quad \{t \mid (\forall u)\big(u \in \emptyset \implies tu \in A\big)\} \quad = \quad \Lambda$$

More generally,

- **in presence possibly empty types interpretations, there is no invariant preserved by $\mathrel{\square\!\!\rightarrow}$.**

We thus have to ensure that $[\![-]\!]$ maps types to **non-empty** sets of terms. In the case of normalization properties, one usually requires $[\![T]\!]$ to contain variables. Similarly as with the weak expansion requirement discussed in §5.3.2, in order to ensure that this invariant is preserved by the (interpretations of) type constructors, we formulate it as follows:

- $E[x] \in [\![T]\!]$ for all $E[\ ] \in \mathrm{Elim}$ and all $x \in \mathcal{X}$.

This invariant is designed to be preserved by the **negative** interpretations $\mathrel{\square\!\!\rightarrow}$ and $\boxtimes$. It is however **not** satisfied by the **positive** interpretations $\mathbf{0}$, $\mathbf{1}$ and $\oplus$. These interpretations could be adapted in order to satisfy our requirement. But we prefer to take the opportunity to define their respective **negative** counterparts $\bot$, $\top$ and $\boxplus$.

Consider the case of $(-) + (-)$. In order to get a negative interpretation, we follow our methodology and look at the elimination rule

$$(\text{+-E})\ \frac{\mathcal{E} \vdash t : T_1 + T_2 \qquad \mathcal{E}, x : T_1 \vdash u_1 : U \qquad \mathcal{E}, x : T_2 \vdash u_2 : U}{\mathcal{E} \vdash \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u_1 \mid \texttt{in}_2\, x \mapsto u_2\} : U}$$

There is a difficulty here: the type U to which `case` eliminates a term $t : T_1 + T_2$ has in general no relation with T_1 nor T_2. A tentative definition of $A \boxplus B$ could be to take the set of all terms t such that

$$\begin{aligned} &\forall T \in \text{Ty}, \\ &\forall u \text{ s.t. } u[w/x] \in [\![T]\!] \text{ whenever } w \in A, \\ &\forall v \text{ s.t. } v[w/x] \in [\![T]\!] \text{ whenever } w \in B, \\ &\qquad\qquad\qquad \text{we have} \quad \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\} \in [\![T]\!] \end{aligned}$$

However, such a definition would not be well-formed since the interpretation $[\![T]\!]$ may itself use $\boxplus$.

The standard way to get rid of this circularity is to define $\boxplus$ by quantifying not over types, but rather over a predefined family of "candidate interpretations". The family presented here is the family of "saturated sets".

Definition 5.15 (Saturated Sets)**.** *The set $\mathcal{SAT}_{\mathcal{WN}}$ of $\mathcal{WN}$-**saturated** sets is the set of all $A \subseteq \mathcal{WN}$ such that*

(i) $E[x] \in A$ *for all* $E[\ ] \in \text{Elim}$, *and*

(ii) $E[t] \in A$ *if* $t \rhd_0 u$ *for some* $E[u] \in A$.

We first check that the family $\mathcal{SAT}_{\mathcal{WN}}$ is not empty.

Lemma 5.16. *The set $\mathcal{SAT}_{\mathcal{WN}}$ is a complete lattice w.r.t. inclusion, whose top element is $\top := \mathcal{WN}$ and whose least element is $\bot := \{t \mid (\exists E[\ ], x)(t \rhd^*_{\text{wh}} E[x])\}$.*

The complete lattice structure of $\mathcal{SAT}_{\mathcal{WN}}$ gives us negative interpretations of `void` and `unit`.

- We take the negative interpretation of `void` to be $\bot$. Note that

  $$\bot \quad = \quad \big\{t \mid (\forall A \in \mathcal{SAT}_{\mathcal{WN}})\big(\texttt{case}_\bot\ t\ \{\} \in A\big)\big\}$$

  Proof. Exercise! □

  Hence, if $t \in \bot$, then we have $\texttt{case}_\bot\ t\ \{\} \in A$ for any $A \in \mathcal{SAT}_{\mathcal{WN}}$. It follows that $[\![\texttt{void}]\!] = \bot$ validates the rule (`void`-E).

- Similarly, we take the negative interpretation of `unit` to be $\top$. It validates (`unit`-I) since $\langle\rangle \in \mathcal{WN}$.

The negative interpretation of $(-) + (-)$ is defined as follows.

Definition 5.17. *Given $A, B \in \mathcal{P}(\Lambda)$, let $A \boxplus_{\mathcal{WN}} B$ be the set of all terms t such that*

$$\begin{aligned}
&\forall C \in \mathcal{SAT}_{\mathcal{WN}},\\
&\forall u \text{ s.t. } u[w/x] \in C \text{ whenever } w \in A,\\
&\forall v \text{ s.t. } v[w/x] \in C \text{ whenever } w \in B,\\
&\qquad\qquad\qquad\qquad \text{we have} \quad \texttt{case}\ t\ \{\texttt{in}_1\, x \mapsto u \mid \texttt{in}_2\, x \mapsto v\} \in C
\end{aligned}$$

Note that $\boxplus_{\mathcal{WN}}$ depends on $\mathcal{SAT}_{\mathcal{WN}}$. Our connectives preserve saturated sets.

Lemma 5.18. *Let $A, B \in \mathcal{SAT}_{\mathcal{WN}}$. Then*

$$A \mathrel{\square\!\!\rightarrow} B,\ A \boxtimes B,\ A \boxplus_{\mathcal{WN}} B \ \in\ \mathcal{SAT}_{\mathcal{WN}}$$

We thus get the analogue of the Adequacy Theorem 5.10.

Theorem 5.19 (Adequacy)**.** *Let $[\![-]\!]$ be a type interpretation such that*

(i) $[\![\kappa]\!] \in \mathcal{SAT}_{\mathcal{WN}}$ *for all* $\kappa \in \mathcal{K}$*, and*

(ii) $[\![\texttt{unit}]\!] = \top$ *and* $[\![\texttt{void}]\!] = \bot$*, and*

(iii) for all types T, U,

$$[\![U \to T]\!] = [\![U]\!] \mathrel{\square\!\!\rightarrow} [\![T]\!] \qquad [\![T \times U]\!] = [\![T]\!] \boxtimes [\![U]\!] \qquad [\![T + U]\!] = [\![T]\!] \boxplus_{\mathcal{WN}} [\![U]\!]$$

Then,

(1) $[\![T]\!] \in \mathcal{SAT}_{\mathcal{WN}}$ *for all type T, and*

(2) $[\![-]\!]$ *is adequate.*

Proof. Exercise! □

The weak normalization of typable terms immediately follows.

Corollary 5.20 (Weak Normalization)**.** *If $\mathcal{E} \vdash t : T$ then $t \in \mathcal{WN}$.*

Proof. Exercise! □

5.5. Strong Normalization

We now extend the material of §5.4 in order to get the **strong normalization** of well-typed terms. We begin in §5.5.1 and §5.5.2 with combinatorial properties on the notion of strong normalization. The (then direct) adaptations to Saturated Sets and Adequacy are discussed in §5.5.3.

5.5.1. Inductive Definition of Strong Normalization

We momentarily step back from λ-terms to Abstract Reduction Systems (ARS's, §3.4).

Let $(A, \rhd)$ be an ARS. We are interested in giving an inductive description of the strongly normalizing elements of A. Recall that according to Def. 3.33 (§3.4.3), $a \in A$ is **strongly normalizing** if there is no infinite sequence $(a_i)_{i \in \mathbb{N}}$ of elements of A such that $a_0 = a$ and $a_i \rhd a_{i+1}$ for all $i \in \mathbb{N}$. In the following it is convenient to say that an **infinite reduction sequence from** $a \in A$ is an infinite sequence $(a_i)_{i \in \mathbb{N}}$ of elements of A such that $a_0 = a$ and $a_i \rhd a_{i+1}$ for all $i \in \mathbb{N}$.

Definition 5.21. *Let $\mathcal{SN}$ be the least subset of A such that for all $a \in A$ we have*

$$(\forall b \in A)\big(a \rhd b \implies b \in \mathcal{SN}\big) \implies a \in \mathcal{SN}$$

Note that $\mathcal{SN}$ can be characterized as the least subset of A closed under the rule

$$\frac{(\forall b \in A)\big(a \rhd b \implies b \in \mathcal{SN}\big)}{a \in \mathcal{SN}}$$

It is direct to see that all $a \in \mathcal{SN}$ are strongly normalizing.

Lemma 5.22. *If $a \in \mathcal{SN}$ then a strongly normalizing.*

PROOF. Exercise! $\square$

Note that the proof of Lem. 5.22 is constructive, since it proves the negative statement "there is **no** infinite reduction sequence [...]".

On the other hand, the converse of Lem. 5.22 is **not** constructive.

Proposition 5.23. *Given an ARS $(A, \rhd)$ and $a \in A$, we have $a \in \mathcal{SN}$ if and only if a is strongly normalizing.*

PROOF. Exercise! $\square$

The proof of Prop. 5.23 relies on classical logic (as we reasoned by contradiction to prove the positive statement "$a \in \mathcal{SN}$") moreover using a form of the Axiom of Choice to ensure the existence of the infinite reduction sequence.[14]

Notation 5.24. *For the remaining of this §5.5, we let $\mathcal{SN}$ stand for strong normalization w.r.t. β-reduction.*

5.5.2. Strong Normalization and Weak Head Reduction

The adaptation of the saturation clauses of Def. 5.15 to strong normalization is straightforward once we know how to connect strong normalization w.r.t. $\rhd_\beta$ with weak head expansion.

The connection is provided by the two following lemmas, and is best expressed by using a notion of tuple reduction.

[14]Actually an instance of **Dependent Choices**, see *e.g.* [TvD88a, Chap. 4, §2.3].

Notation 5.25. *Write $(t_1, \dots, t_n) \rhd_\beta (u_1, \dots, u_n)$ when there is some $i \in \{1, \dots, n\}$ such that $t_i \rhd_\beta u_i$ and $t_j = u_j$ for all $j \neq i$.*

Let us explain the idea in the case of the pure λ-calculus. We would like to adapt Def. 5.15 to strong normalization. First, for a set $A \in \mathcal{P}(\Lambda)$ to be saturated, we shall ask that A contains only strongly normalizing terms: $A \subseteq \mathcal{SN}$. Now, we have to adapt the two clauses Def. 5.15, in order to avoid conflict with the invariant $A \subseteq \mathcal{SN}$. The first clause

- $E[x] \in A$ for all $E[\ ] \in \text{Elim}$

is obviously adapted to

- $E[x] \in A$ for all $E[\ ] \in \text{Elim} \cap \mathcal{SN}$.

Consider now the second clause:

- $E[(\lambda x.t)u] \in A$ if $E[t[u/x]] \in A$.

This clause is problematic only in the following case: Assume that the variable x does not occur free in t and that u is not normalizable. Then $E[t[u/x]] = E[t]$ can be strongly normalizing while $E[(\lambda x.t)u]$ is not. This is easily patched by asking u to be strongly normalizing:

- $E[(\lambda x.t)u] \in A$ if $E[t[u/x]] \in A$ and $u \in \mathcal{SN}$.

This clause is correct, in the sense that $E[(\lambda x.t)u]$ is strongly normalizing if both $E[t[u/x]]$ and u are.

Let us look at this more precisely. Assume that $E[t[u/x]] \in \mathcal{SN}$ and $u \in \mathcal{SN}$. According to the definition of $\mathcal{SN}$, we must show that for all v, if $E[(\lambda x.t)u] \rhd_\beta v$ then $v \in \mathcal{SN}$. Now, since we are in the pure λ-calculus, $E[\ ]$ is of the form $[\ ]t_1 \dots t_n$. Note that $t, u, t_1, \dots, t_n \in \mathcal{SN}$. The different possibilities for v can be depicted as follows:

$$\begin{tikzcd}
(\lambda x.t)ut_1 \dots t_n \arrow[r, "\rhd_0"] \arrow[d, "\rhd_\beta"'] & t[u/x]t_1 \dots t_n \arrow[d, dotted, "\rhd^*_\beta"] \\
(\lambda x.t')u't'_1 \dots t'_n \arrow[r, dotted, "\rhd_0"'] & t'[u'/x]t'_1 \dots t'_n
\end{tikzcd}$$

Hence, there are schematically two possibilities for v. If $v = t[u/x]t_1 \dots t_n$ then we are done. Otherwise, v is of the form $(\lambda x.t')u't'_1 \dots t'_n$ with $(t, u, t_1, \dots, t_n) \rhd_\beta (t', u', t'_1, \dots, t'_n)$. In this case, we still have $t', u', t'_1, \dots, t'_n \in \mathcal{SN}$ and $t'[u'/x]t'_1 \dots t'_n \in \mathcal{SN}$. This suggests to reason by induction on $t, u, t_1, \dots, t_n \in \mathcal{SN}$.

More generally, we can show by induction on $E[\ ], t, u \in \mathcal{SN}$ that

- $E[(\lambda x.t)u] \in \mathcal{SN}$ if $E[t[u/x]] \in \mathcal{SN}$ and $u \in \mathcal{SN}$.

This reasoning easily generalizes to the other term constructors, as stated in the two following lemmas.

Lemma 5.26 (Weak Standardization). *Let $E[\,] \in \mathrm{Elim}$.*

(1) If $E[(\lambda x.t)u] \rhd_\beta v$, then either $v = E[t[u/x]]$ or v is of the form $E'[(\lambda x.t')u']$ with $(E[\,], t, u) \rhd_\beta (E'[\,], t', u')$.

(2) Let $i \in \{1,2\}$. If $E[\pi_i\langle t_1, t_2\rangle] \rhd_\beta v$, then either $v = E[t_i]$ or v is of the form $E'[\pi_i\langle t'_1, t'_2\rangle]$ with $(E[\,], t_1, t_2) \rhd_\beta (E'[\,], t'_1, t'_2)$.

(3) Let $i \in \{1,2\}$. If $E[\mathtt{case}\ (\mathtt{in}_i\ t)\ \{\mathtt{in}_1\, x \mapsto u_1 \mid \mathtt{in}_2\, x \mapsto u_2\}] \rhd_\beta v$, then either $v = E[u_i[t/x]]$ or v is of the form $E[\mathtt{case}\ (\mathtt{in}_i\ t')\ \{\mathtt{in}_1\, x \mapsto u'_1 \mid \mathtt{in}_2\, x \mapsto u'_2\}]$ with $(E[\,], t, u_1, u_2) \rhd_\beta (E[\,], t', u'_1, u'_2)$.

Proof. Exercise! $\square$

The Weak Standardization Lemma 5.26 exhibits a very good behaviour of weak head expansion w.r.t. strong normalization.

Lemma 5.27 (Weak Head Expansion). *Let $E[\,] \in \mathrm{Elim}$.*

(1) If $E[t[u/x]] \in \mathcal{SN}$ and $u \in \mathcal{SN}$ then $E[(\lambda x.t)u] \in \mathcal{SN}$.

(2) Let $i \in \{1,2\}$. If $E[t_i] \in \mathcal{SN}$ and $t_{3-i} \in \mathcal{SN}$ then $E[\pi_i\langle t_1, t_2\rangle] \in \mathcal{SN}$.

(3) Let $i \in \{1,2\}$. If $E[u_i[t/x]] \in \mathcal{SN}$ and $t, u_{3-i} \in \mathcal{SN}$ then $E[\mathtt{case}\ (\mathtt{in}_i\ t)\ \{\mathtt{in}_1\, x \mapsto u_1 \mid \mathtt{in}_2 \mapsto u_2\}] \in \mathcal{SN}$.

Proof. Exercise! $\square$

5.5.3. Type Interpretations and Adequacy

We can now give the adaptation of Def. 5.15 to strong normalization.

Definition 5.28 (Saturated Sets). *The set $\mathcal{SAT}_{\mathcal{SN}}$ of $\mathcal{SN}$-**saturated** sets is the set of all $A \subseteq \mathcal{SN}$ such that*

(i) $E[x] \in A$ whenever $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$, and

(ii) $E[(\lambda x.t)u] \in A$ whenever $u \in \mathcal{SN}$ and $E[t[u/x]] \in A$, and

(iii) for all $i \in \{1,2\}$, $E[\pi_i\langle t_1, t_2\rangle] \in A$ whenever $t_{3-i} \in \mathcal{SN}$ and $E[t_i] \in A$, and

(iv) for all $i \in \{1,2\}$, $E[\mathtt{case}\ (\mathtt{in}_i\ t)\ \{\mathtt{in}_1\, x \mapsto u_1 \mid \mathtt{in}_2\, x \mapsto u_2\}] \in A$ whenever $t, u_{3-i} \in \mathcal{SN}$ and $E[u_i[t/x]] \in A$.

The non-degeneracy of Def. 5.28 directly follows from Lem. 5.27. This leads to the analogue of Lem. 5.16.

Lemma 5.29. *The set $\mathcal{SAT}_{\mathcal{SN}}$ is a complete lattice w.r.t. inclusion, whose top element is $\top := \mathcal{SN}$ and whose least element is $\bot := \{t \in \mathcal{SN} \mid (\exists E[\,], x)\big(t \rhd^*_{\mathrm{wh}} E[x]\big)\}$.*

Proof. Exercise! $\square$

Recall that the negative interpretation of $(-) + (-)$ depends on the family of candidate interpretations over which we are quantifying. It has thus to be adapted. Given $A, B \in \mathcal{P}(\Lambda)$, let $A \boxplus_{\mathcal{SN}} B$ be the set of all terms t such that

$$\begin{array}{l} \forall C \in \mathcal{SAT}_{\mathcal{SN}}, \\ \forall u \text{ s.t. } u[w/x] \in C \text{ whenever } w \in A, \\ \forall v \text{ s.t. } v[w/x] \in C \text{ whenever } w \in B, \\ \qquad\qquad\qquad\qquad \text{we have} \quad \mathtt{case}\ t\ \{\mathtt{in}_1\, x \mapsto u \mid \mathtt{in}_2\, x \mapsto v\} \in C \end{array}$$

Lemma 5.30. *Let $A, B \in \mathcal{SAT}_{\mathcal{SN}}$. Then*

$$A \mathrel{\square\!\!\rightarrow} B,\ A \boxtimes B,\ A \boxplus_{\mathcal{SN}} B \ \in\ \mathcal{SAT}_{\mathcal{SN}}$$

Proof. Exercise! □

We thus get the analogue of Adequacy Theorem 5.19.

Theorem 5.31 (Adequacy). *Let $[\![-]\!]$ be a type interpretation such that*

(i) $[\![\kappa]\!] \in \mathcal{SAT}_{\mathcal{SN}}$ *for all* $\kappa \in \mathcal{K}$, *and*

(ii) $[\![\mathtt{unit}]\!] = \top$ *and* $[\![\mathtt{void}]\!] = \bot$, *and*

(iii) for all types T, U,

$$[\![U \to T]\!] = [\![U]\!] \mathrel{\square\!\!\rightarrow} [\![T]\!] \qquad [\![T \times U]\!] = [\![T]\!] \boxtimes [\![U]\!] \qquad [\![T + U]\!] = [\![T]\!] \boxplus_{\mathcal{SN}} [\![U]\!]$$

Then $[\![-]\!]$ is adequate.

Proof. Exercise! □

We finally obtain the strong normalization of typable terms (Thm. 4.12, §4.2.2 and Thm. 4.31, §4.4.4).

Theorem 5.32 (Strong Normalization). *If $\mathcal{E} \vdash t : T$ then $t \in \mathcal{SN}$.*

Proof. Exercise! □

6. First-Order Predicate Logic

We now turn to first-order predicate logic.

The status of the Curry-Howard correspondence for intuitionistic first-order logic is somehow ambiguous. On the one hand, everything works (almost) perfectly well at the technical level. On the other hand, in proofs-assistants based on the Curry-Howard correspondence, the setting of **dependent types** is often preferred to first-order logic (see [SU06, §13] or [Uni13, §1], also Rem. 6.48, §6.3).[15]

[15] See also [Jac01, Hof97] (beyond the scope of this course).

We shall pay special attention to the specific properties of **intuitionistic** first-order predicate logic (NJ_1), most notably concerning the extraction of witnesses from proofs of existential statements. Further, the already rich setting of NJ_1 gives basic intuitions on constructive reasoning, and is the basis of important stronger systems, such as Intuitionistic Arithmetic (7.3) and Second-Order Logic (**??**). Compared with dependent types, one feature of first-order logic is that it is sufficiently simple for easily stating (and proving) interesting non-trivial meta-theoretical properties (§7.3.3, §7.3.4, §8.3).

As shall see, the Curry-Howard correspondence for intuitionistic first-order logic is based on proof-terms which are in essence simply-typed, so that in particular their normalization is no more difficult than in the case of propositional logic (§5, §6.5). Besides, while they are not particularly interesting from a programming point of view, the proof-terms for NJ_1 will play a crucial role in establishing the main extraction properties of the latter.

We assume a working knowledge of first-order logic (essentially covered by [vD04]). We however recall the relevant definitions and results.

6.1. Preliminaries

This §6.1 gathers definitions on **many-sorted** first-order logic. It essentially consists of a mild extension of [vD04, §2.2 & §2.3] (see also [SU06, §8.1 & §8.4] and [Bus98a, §2.1]).

6.1.1. First-Order Signatures

Definition 6.1 (First-Order Signature)**.** *A (first-order)* ***signature*** $\mathbf{\Sigma}$ *consists of*

(i) a set $\mathrm{Sort}(\mathbf{\Sigma})$ *of* ***sorts*** *(ranged over by* $\sigma, \tau, \theta, \dots$*);*

(ii) a collection of (sorted) ***function symbols***

$$\mathrm{Fun}(\mathbf{\Sigma}) \quad = \quad \bigcup_{(\tau_1,\dots,\tau_n;\sigma)\in\mathrm{Sort}(\mathbf{\Sigma})^{n+1}} \mathrm{Fun}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n;\sigma)$$

(we sometimes write $\mathsf{f} : \tau_1 \times \dots \times \tau_n \to \sigma$ *for* $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n;\sigma)$*);*

(iii) a collection of (sorted) ***predicate*** *(or* ***relation****) symbols*

$$\mathrm{Pred}(\mathbf{\Sigma}) \quad = \quad \bigcup_{(\tau_1,\dots,\tau_n)\in\mathrm{Sort}(\mathbf{\Sigma})^{n}} \mathrm{Pred}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n)$$

(we sometimes write $\mathsf{P} \subseteq \tau_1 \times \dots \times \tau_n$ *for* $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n)$*).*

Notation 6.2. *Functions symbols* $\mathsf{c} \in \mathrm{Fun}(\mathbf{\Sigma})(;\sigma)$ *(i.e.* $\mathrm{Fun}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n;\sigma)$ *with* $n = 0$*) are often called* ***constants****. We write* $\mathsf{c} : \sigma$ *instead of* $\mathsf{c} : \mathbf{1} \to \sigma$ *(i.e.* $\tau_1 \times \dots \times \tau_n \to \sigma$ *with* $n = 0$*).*

Definition 6.3 (First-Order Terms)**.** *Let* $\mathbf{\Sigma}$ *be a signature. Assume given a collection* $\mathcal{V} = \bigcup_{\sigma\in\mathrm{Sort}(\mathbf{\Sigma})} \mathcal{V}(\sigma)$ *of* $\mathbf{\Sigma}$***-sorted first-order variables****, where* $\mathcal{V}(\sigma)$ *(also written* $\mathcal{V}^\sigma$*) is ranged over by* $\mathsf{x}^\sigma, \mathsf{y}^\sigma, \mathsf{z}^\sigma$*, etc.*

*(1) The set of **first-order terms** over* $(\mathbf{\Sigma},\mathcal{V})$ *is the collection* $\mathrm{Ter}(\mathbf{\Sigma},\mathcal{V}) = \bigcup_{\sigma\in\mathrm{Sort}(\mathbf{\Sigma})}\mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\sigma)$ *inductively defined as follows:*

- *for each* $\mathsf{x}^\sigma \in \mathcal{V}(\sigma)$*, we have* $\mathsf{x}^\sigma \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\sigma)$*;*
- *given* $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n;\sigma)$ *and* $\mathsf{a}_1 \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\tau_1),\dots,\mathsf{a}_n \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\tau_n)$*, we have* $\mathsf{f}(\mathsf{a}_1,\dots,\mathsf{a}_n) \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\sigma)$*.*

(2) The set $\mathrm{FV}(\mathsf{a})$ *is defined by induction on* a *as follows:*

- $\mathrm{FV}(\mathsf{x}^\sigma) = \{x^\sigma\}$*;*
- $\mathrm{FV}(\mathsf{f}(\mathsf{a}_1,\dots,\mathsf{a}_n)) = \mathrm{FV}(\mathsf{a}_1)\cup\dots\cup\mathrm{FV}(\mathsf{a}_n)$*.*

Notation 6.4.

(1) When $\mathbf{\Sigma}$ *and* $\mathcal{V}$ *are understood from the context, we often write* a^σ *for* $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\sigma)$*.*

(2) Given terms a^σ *and* b *and a variable* x^σ*, the **substitution** of* x^σ *by* a^σ *in* b*, notation* $\mathsf{b}[\mathsf{a}^\sigma/\mathsf{x}^\sigma]$*, is defined in the obvious way.*

(3) When $\mathbf{\Sigma}$ *is one-sorted, i.e. when* $\mathrm{Sort}(\mathbf{\Sigma})$ *is a singleton (say* $\{\sigma\}$*), we write*

- $\mathrm{Fun}(\mathbf{\Sigma})(n)$ *for* $\mathrm{Fun}(\mathbf{\Sigma})(\underbrace{\sigma,\dots,\sigma}_{n};\sigma)$*;*
- $\mathrm{Pred}(\mathbf{\Sigma})(n)$ *for* $\mathrm{Pred}(\mathbf{\Sigma})(\underbrace{\sigma,\dots,\sigma}_{n})$*;*
- $\mathsf{x} \in \mathcal{V}$ *for* $\mathsf{x}^\sigma \in \mathcal{V}(\sigma)$*;*
- $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})$ *for* $\mathsf{a}^\sigma \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\sigma)$*.*

6.1.2. The Language of First-Order Predicate Logic

Let $\mathbf{\Sigma}$ be a first-order signature and $\mathcal{V}$ be a collection of $\mathbf{\Sigma}$-sorted first-order variables. The formulae of first-order predicate logic over $(\mathbf{\Sigma},\mathcal{V})$ are given by the grammar:

$$\begin{array}{rcl} A, B & ::= & \mathsf{P}(\mathsf{a}_1,\dots,\mathsf{a}_n) \;\mid\; A \Rightarrow B \;\mid\; A\wedge B \;\mid\; A \vee B \;\mid\; \top \;\mid\; \bot \\ & \mid & (\forall \mathsf{x}^\sigma)A \;\mid\; (\exists \mathsf{x}^\sigma)A \end{array}$$

where $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n)$ and $\mathsf{a}_i \in \mathrm{Ter}(\mathbf{\Sigma},\mathcal{V})(\tau_i)$ for each $i = 1,\dots,n$.

The set $\mathrm{FV}(A)$ of **free variables** of A is defined by induction on A as follows:

$$\begin{array}{rcl@{\qquad}rcl} \mathrm{FV}(\mathsf{P}(\mathsf{a}_1,\dots,\mathsf{a}_n)) & := & \bigcup_{1\le i\le n}\mathrm{FV}(\mathsf{a}_i) & \mathrm{FV}(A\Rightarrow B) & := & \mathrm{FV}(A)\cup\mathrm{FV}(B) \\ \mathrm{FV}((\forall\mathsf{x}^\sigma)A) & := & \mathrm{FV}(A)\setminus\{\mathsf{x}^\sigma\} & \mathrm{FV}(A\wedge B) & := & \mathrm{FV}(A)\cup\mathrm{FV}(B) \\ \mathrm{FV}((\exists\mathsf{x}^\sigma)A) & := & \mathrm{FV}(A)\setminus\{\mathsf{x}^\sigma\} & \mathrm{FV}(A\vee B) & := & \mathrm{FV}(A)\cup\mathrm{FV}(B) \\ & & & \mathrm{FV}(\top) & := & \emptyset \\ & & & \mathrm{FV}(\bot) & := & \emptyset \end{array}$$

Notation 6.5.

(1) A closed formula (i.e. a formula A *such that* $\mathrm{FV}(A) = \emptyset$*) is often called a **sentence**.*

(2) *When $\mathbf{\Sigma}$ is one-sorted, i.e. when* $\mathrm{Sort}(\mathbf{\Sigma})$ *is a singleton (say* $\{\sigma\}$*), we write* $(\forall \mathsf{x})$ *and* $(\exists \mathsf{x})$ *for resp.* $(\forall \mathsf{x}^\sigma)$ *and* $(\exists \mathsf{x}^\sigma)$*.*

(3) *We often write* $A(\mathsf{x}^\sigma)$ *to* ***informally*** *emphasize that the variable* x^σ *occurs or may occur free in the formula* A *(and similarly for first-order terms).*

Warning 6.6 (Variable Binding)**.** *Universal quantifiers* $(\forall \mathsf{x}^\sigma)A$ *("for all") and existential quantifiers* $(\exists \mathsf{x}^\sigma)A$ *("there exists")* ***bind*** x^σ *in* A*. We thus assume that first-order formulae are quotiented by* α*-equivalence (w.r.t. the obvious adaptation of* §3.2.4*).*

Given a term b^σ and a variable x^σ the **capture-avoiding substitution** of x^σ by b^σ in A is defined by induction on A as follows:

$$\begin{array}{rcll}
\top[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & \top & \\
\bot[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & \bot & \\
\big(\mathsf{P}(\mathsf{a}_1,\dots,\mathsf{a}_n)\big)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & \mathsf{P}\big(\mathsf{a}_1[\mathsf{b}^\sigma/\mathsf{x}^\sigma],\dots,\mathsf{a}_n[\mathsf{b}^\sigma/\mathsf{x}^\sigma]\big) & \\
(A \Rightarrow B)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & A[\mathsf{b}^\sigma/\mathsf{x}^\sigma] \;\Rightarrow\; B[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & \\
(A \wedge B)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & A[\mathsf{b}^\sigma/\mathsf{x}^\sigma] \;\wedge\; B[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & \\
(A \vee B)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & A[\mathsf{b}^\sigma/\mathsf{x}^\sigma] \;\vee\; B[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & \\
\big((\forall \mathsf{y}^\tau)A\big)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & (\forall \mathsf{y}^\tau)\big(A[\mathsf{b}^\sigma/\mathsf{x}^\sigma]\big) & \text{if } \mathsf{y} \neq \mathsf{x} \text{ and } \mathsf{y} \notin \mathrm{FV}(\mathsf{b}) \\
\big((\exists \mathsf{y}^\tau)A\big)[\mathsf{b}^\sigma/\mathsf{x}^\sigma] & := & (\exists \mathsf{y}^\tau)\big(A[\mathsf{b}^\sigma/\mathsf{x}^\sigma]\big) & \text{if } \mathsf{y} \neq \mathsf{x} \text{ and } \mathsf{y} \notin \mathrm{FV}(\mathsf{b})
\end{array}$$

Notation 6.7. *We extend the notations of* §2 *with the convention that quantifiers* ($(\forall x^\sigma)$ *and* $(\exists x^\sigma)$) *have the highest priority, so that e.g.* $(\exists x^\sigma)A \wedge (\forall y^\tau)B \Rightarrow C$ *stands for* $\big(((\exists x^\sigma)A) \wedge ((\forall y^\tau)B)\big) \Rightarrow C$*.*

6.1.3. Structures and Models

Definition 6.8. *A* ***model*** *of a signature* $\mathbf{\Sigma}$ *(or* $\mathbf{\Sigma}$***-structure****)* $\mathcal{M}$ *consists of*

(i) *for each sort* $\sigma \in \mathrm{Sort}(\mathbf{\Sigma})$*, a* ***non-empty*** *set* $\mathcal{M}(\sigma)$ *(also written* $\mathcal{M}^\sigma$*, or even* $[\![\sigma]\!]$ *when* $\mathcal{M}$ *is clear from the context);*

(ii) *for each function symbol* $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n;\sigma)$*, a function* $\mathcal{M}(\mathsf{f}) : \mathcal{M}(\tau_1) \times \dots \times \mathcal{M}(\tau_n) \to \mathcal{M}(\sigma)$ *(we also write* $\mathsf{f}_\mathcal{M}$ *or* $\mathsf{f}^\mathcal{M}$ *for* $\mathcal{M}(\mathsf{f})$*, or even* $[\![\mathsf{f}]\!]$ *when* $\mathcal{M}$ *is clear from the context);*

(iii) *for each predicate symbol* $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\tau_1,\dots,\tau_n)$*, a subset* $\mathcal{M}(\mathsf{P})$ *of* $\mathcal{M}(\tau_1) \times \dots \times \mathcal{M}(\tau_n)$ *(we also write* $\mathsf{P}_\mathcal{M}$ *or* $\mathsf{P}^\mathcal{M}$ *for* $\mathcal{M}(\mathsf{P})$*, or even* $[\![\mathsf{P}]\!]$ *when* $\mathcal{M}$ *is clear from the context).*

Remark 6.9. *A constant* $\mathsf{c} \in \mathrm{Fun}(\mathbf{\Sigma})(;\sigma)$ *is interpreted as a function* $\mathcal{M}(\mathsf{c}) : \mathbf{1} \to \mathcal{M}(\sigma)$ *(where* $\mathbf{1} = \{\bullet\}$*), equivalently as an element* $\mathcal{M}(\mathsf{c})(\bullet) \in \mathcal{M}(\sigma)$ *that we still denote* $\mathcal{M}(\mathsf{c})$*.*

Notation 6.10. *When* $\mathbf{\Sigma}$ *is one-sorted, i.e. when* $\mathrm{Sort}(\mathbf{\Sigma})$ *is a singleton (say* $\{\sigma\}$*), we write* $\mathcal{M}$ *for* $\mathcal{M}(\sigma)$*.*

$$
\begin{array}{lll}
\mathcal{M}, v \models \mathsf{P}(\mathsf{a}_1, \dots, \mathsf{a}_n) & \text{iff} & ([\![\mathsf{a}_1]\!]v, \dots, [\![\mathsf{a}_n]\!]v) \in [\![\mathsf{P}]\!] \\
\mathcal{M}, v \models A \wedge B & \text{iff} & \mathcal{M}, v \models A \text{ and } \mathcal{M}, v \models B \\
\mathcal{M}, v \models A \vee B & \text{iff} & \mathcal{M}, v \models A \text{ or } \mathcal{M}, v \models B \\
\mathcal{M}, v \models A \Rightarrow B & \text{iff} & \mathcal{M}, v \models A \text{ implies } \mathcal{M}, v \models B \\
\mathcal{M}, v \models (\forall \mathsf{x}^\sigma)A & \text{iff} & \text{for all } a \in \mathcal{M}(\sigma),\ \mathcal{M}, v[a/\mathsf{x}^\sigma] \models A \\
\mathcal{M}, v \models (\exists \mathsf{x}^\sigma)A & \text{iff} & \text{there exists } a \in \mathcal{M}(\sigma) \text{ such that } \mathcal{M}, v[a/\mathsf{x}^\sigma] \models A \\
\mathcal{M}, v \models \top & & \\
\mathcal{M}, v \not\models \bot & &
\end{array}
$$

Figure 9: The Satisfaction Relation $\mathcal{M}, v \models A$.

The notion of satisfaction of a formula A in a model $\mathcal{M}$ under a valuation v (notation $\mathcal{M}, v \models A$) is standard. First, by a **valuation** v we mean a collection $(v^\sigma)_{\sigma \in \mathrm{Sort}(\mathbf{\Sigma})}$ where $v^\sigma : \mathcal{V}(\sigma) \to \mathcal{M}(\sigma)$. As usual, given x^σ and $a \in \mathcal{M}(\sigma)$, we let $v[a/\mathsf{x}^\sigma]$ be the valuation which takes x^σ to a and which is equal to v everywhere else. Given a valuation v, each first-order term $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})(\sigma)$ induces an individual $[\![\mathsf{a}]\!]v \in \mathcal{M}(\sigma)$ in the obvious way.

Definition 6.11. *Let A be a first-order formula over a first-order $\mathbf{\Sigma}$ be a signature.*

*(1) Given a $\mathbf{\Sigma}$-structure $\mathcal{M}$ and a valuation v, we say that A is **satisfied in $\mathcal{M}$ under** v if $\mathcal{M}, v \models A$, where the relation $\mathcal{M}, v \models A$ is defined in Fig. 9.*

*(2) Given a $\mathbf{\Sigma}$-structure $\mathcal{M}$, we say that A is **satisfiable in** $\mathcal{M}$ if $\mathcal{M}, v \models A$ for some valuation v.*

*We say that A is **satisfiable** if A is satisfiable in some $\mathbf{\Sigma}$-structure $\mathcal{M}$.*

*(3) Given a $\mathbf{\Sigma}$-structure $\mathcal{M}$, we say that A is **valid in** $\mathcal{M}$ (notation $\mathcal{M} \models A$) if A is satisfied in $\mathcal{M}$ under all valuations v.*

*We say that A is **valid** (notation $\models A$) if A is valid in all $\mathbf{\Sigma}$-structures $\mathcal{M}$.*

Example 6.12. *Assume that x^σ is not free in B. Then the following formulae are valid:*

(1) $\big((\forall \mathsf{x}^\sigma)(A(\mathsf{x}^\sigma) \Rightarrow B)\big) \Leftrightarrow \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big)$

(2) $\big((\exists \mathsf{x}^\sigma)(A(\mathsf{x}^\sigma) \Rightarrow B)\big) \Leftrightarrow \big((\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big)$

6.2. Natural Deduction for First-Order Predicate Logic

Recall from§2.1 (resp. §2.3) the system NJ_0 (resp. NK_0) of natural deduction for intuitionistic (resp. classical) propositional logic. Natural deduction for intuitionistic (resp. classical) first-order predicate logic is given by extending NJ_0 (resp. NK_0) with rules for universal ($\forall$) and existential ($\exists$) quantifiers.

Formally, fix a first-order signature $\mathbf{\Sigma}$ and a set $\mathcal{V}$ of $\mathbf{\Sigma}$-sorted first-order variables. Natural deduction for **intuitionistic** first-order predicate logic (NJ_1) over $(\mathbf{\Sigma}, \mathcal{V})$ is given by the deduction rules for Fig. 1, where given $\Delta = A_1, \dots, A_n$, we write $\mathrm{FV}(\Delta)$ for

$$\text{(Ax)}\ \frac{}{\Delta \vdash A}\ (A \in \Delta) \qquad (\top\text{-I})\ \frac{}{\Delta \vdash \top} \qquad (\bot\text{-E})\ \frac{\Delta \vdash \bot}{\Delta \vdash A}$$

$$(\Rightarrow\text{-I})\ \frac{\Delta, A \vdash B}{\Delta \vdash A \Rightarrow B} \qquad (\Rightarrow\text{-E})\ \frac{\Delta \vdash A \Rightarrow B \qquad \Delta \vdash A}{\Delta \vdash B}$$

$$(\wedge\text{-I})\ \frac{\Delta \vdash A \qquad \Delta \vdash B}{\Delta \vdash A \wedge B} \qquad (\wedge_1\text{-E})\ \frac{\Delta \vdash A \wedge B}{\Delta \vdash A} \qquad (\wedge_2\text{-E})\ \frac{\Delta \vdash A \wedge B}{\Delta \vdash B}$$

$$(\vee_1\text{-I})\ \frac{\Delta \vdash A}{\Delta \vdash A \vee B} \qquad (\vee_2\text{-I})\ \frac{\Delta \vdash B}{\Delta \vdash A \vee B}$$

$$(\vee\text{-E})\ \frac{\Delta \vdash A \vee B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Delta \vdash C}$$

$$(\forall\text{-I})\ \frac{\Delta \vdash A}{\Delta \vdash (\forall \mathsf{x}^\sigma)A}(\mathsf{x}^\sigma \notin \mathrm{FV}(\Delta)) \qquad (\forall\text{-E})\ \frac{\Delta \vdash (\forall \mathsf{x}^\sigma)A}{\Delta \vdash A[\mathsf{a}^\sigma/\mathsf{x}^\sigma]}$$

$$(\exists\text{-I})\ \frac{\Delta \vdash A[\mathsf{a}^\sigma/\mathsf{x}^\sigma]}{\Delta \vdash (\exists \mathsf{x}^\sigma)A} \qquad (\exists\text{-E})\ \frac{\Delta \vdash (\exists \mathsf{x}^\sigma)A \qquad \Delta, A \vdash B}{\Delta \vdash B}\ (\mathsf{x}^\sigma \notin \mathrm{FV}(\Delta, B))$$

Figure 10: Natural Deduction Rules for Intuitionistic First-Order Predicate Logic.

$\mathrm{FV}(A_1) \cup \cdots \cup \mathrm{FV}(A_n)$. Natural deduction for **classical** first-order predicate logic (NK_1) over $(\mathbf{\Sigma}, \mathcal{V})$ is NJ_1 augmented with the excluded middle (EM) (Def. 2.19, §2.2.1).

Remark 6.13. *Similarly as in §2.1 (Rem. 2.8) we shall* ***informally*** *speak of intuitionistic and classical predicate* ***logics*** *for the* ***deduction systems*** *NJ_1 and NK_1. We again insist on the fact that there are different possible deduction systems for these logics.*

The remaining of this §6.2 is organized as follows. We state in §6.2.1 the expected usual easy structural properties of NJ_1 and NK_1. The main results relating syntax and semantics of (classical) first-order logic are surveyed in §6.2.2. In §6.2.3 we briefly indicate the extensions to first-order logic of the negative translations from §2.5. The undecidability of provability in NJ_1 and NK_1 is proven in §6.2.4, using material from §3.4.1 and §3.7 on the untyped λ-calculus and Combinatory Logic. The failure of the extension of Glivenko's Theorem 2.48 (§2.6) to the full language of first-order logic is discussed in §6.2.5.

Last but not least, the distinctive properties of NJ_1 over NK_1, namely the Disjunction Property and the analogous **Witness Property** for existential quantifications, are stated in §6.2.6, proved in §6.3.3 and further discussed in §6.4.6.

But before, let us look at some examples!

Example 6.14. *In NJ_1 we have $\vdash (\forall \mathsf{x}^\sigma)A \Rightarrow (\exists \mathsf{x}^\sigma)A$ with the proof tree*

In Intuitionistic First-Order Predicate Logic (NJ_1):

$$\begin{array}{lrcl}
\vdash & \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big) & \Leftrightarrow & (\forall \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big) \\
\vdash & \neg(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) & \Leftrightarrow & (\forall \mathsf{x}^\sigma)\neg A(\mathsf{x}^\sigma) \\
\vdash & \neg\neg(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) & \Leftrightarrow & \neg(\forall \mathsf{x}^\sigma)\neg A(\mathsf{x}^\sigma) \\
\\
\vdash & (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big) & \Rightarrow & (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B \\
\vdash & (\exists \mathsf{x}^\sigma)\neg A(\mathsf{x}^\sigma) & \Rightarrow & \neg(\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \\
\vdash & (\exists \mathsf{x}^\sigma)\neg\neg A(\mathsf{x}^\sigma) & \Rightarrow & \neg\neg(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)
\end{array}$$

In Classical First-Order Predicate Logic (NK_1):

$$\vdash \quad (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big) \quad \Leftrightarrow \quad \big((\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big)$$

Table 3: Some Laws of First-Order Predicate Logic, where $\mathsf{x}^\sigma \notin \mathrm{FV}(B)$.

$$\cfrac{\cfrac{\overline{(\forall \mathsf{x}^\sigma)A \vdash (\forall \mathsf{x}^\sigma)A}}{(\forall \mathsf{x}^\sigma)A \vdash A}}{(\forall \mathsf{x}^\sigma)A \vdash (\exists \mathsf{x}^\sigma)A}$$

Exercise 6.15. *Show that the following, where $\mathsf{x}^\sigma \notin \mathrm{FV}(B)$, are derivable in NJ_1:*

(1) $\vdash (\forall \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big) \Rightarrow \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big)$

(2) $\vdash \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big) \Rightarrow (\forall \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big)$

(3) $\vdash (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \Rightarrow B\big) \Rightarrow \big((\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow B\big)$

Exercise 6.16. *Assume that x^σ is not free in B. Show that in NJ_1 we have*

$$\vdash \quad \big((\forall \mathsf{x}^\sigma)\neg A(\mathsf{x}^\sigma) \Rightarrow B\big) \quad \Rightarrow \quad \neg\neg(\exists \mathsf{x}^\sigma)\big(\neg A(\mathsf{x}^\sigma) \Rightarrow B\big)$$

Example 6.17. *Combining Ex. 6.15 and Ex. 6.16, we obtain the laws summarized in Table 3. In particular, NK_1 proves the following well-known **Drinker's Paradox**:*

$$\vdash \quad (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \quad \Rightarrow \quad (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big)$$

Table 4 displays some further usual laws (see e.g. [vD04, Lem. 5.2.1]).

Exercise 6.18. *Show that in NJ_1 we have* $\vdash \neg\neg(\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \Rightarrow (\forall \mathsf{x}^\sigma)\neg\neg A(\mathsf{x}^\sigma)$

In Intuitionistic First-Order Predicate Logic (NJ_1):

$$\begin{array}{lll}
\vdash\ (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \lor B(\mathsf{x}^\sigma)\big) & \Leftrightarrow & \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\ \lor\ (\exists \mathsf{x}^\sigma)B(\mathsf{x}^\sigma)\big) \\
\vdash\ (\exists \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \land C\big) & \Leftrightarrow & \big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\ \land\ C\big) \\
\vdash\ (\exists \mathsf{x}^\sigma)\big(C \Rightarrow A(\mathsf{x}^\sigma)\big) & \Rightarrow & \big(C \Rightarrow (\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big) \\
 & & \\
\vdash\ (\forall \mathsf{x}^\sigma)\big(C \Rightarrow A(\mathsf{x}^\sigma)\big) & \Leftrightarrow & \big(C \Rightarrow (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big) \\
 & & \\
\vdash\ (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\ \land\ (\forall \mathsf{x}^\sigma)B(\mathsf{x}^\sigma) & \Leftrightarrow & (\forall \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \land B(\mathsf{x}^\sigma)\big) \\
\vdash\ (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\ \lor\ C & \Rightarrow & (\forall \mathsf{x}^\sigma)\big(A(\mathsf{x}^\sigma) \lor C\big)
\end{array}$$

In Classical First-Order Predicate Logic (NK_1):

$$\begin{array}{lll}
\vdash\ (\exists \mathsf{x}^\sigma)\big(C \Rightarrow A(\mathsf{x}^\sigma)\big) & \Leftrightarrow & \big(C\ \Rightarrow\ (\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big) \\
\vdash\ (\forall \mathsf{x}^\sigma)\big(C \lor A(\mathsf{x}^\sigma)\big) & \Leftrightarrow & \big(C\ \lor\ (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big)
\end{array}$$

Table 4: Further Laws of First-Order Predicate Logic, where $\mathsf{x}^\sigma \notin \mathrm{FV}(C)$.

6.2.1. Structural Properties

Similarly as for NJ_0, the following basic properties are proved by induction on derivations.

Lemma 6.19 (Structural Rules)**.** *The following properties hold in* NJ_1 *and* NK_1*:*

(weakening) *if* $\Delta \vdash A$ *then* $\Delta, B \vdash A$*;*

(contraction) *if* $\Delta, B, B \vdash A$ *then* $\Delta, B \vdash A$*;*

(exchange) *if* $\Delta, B, \Delta', C, \Delta'' \vdash A$ *then* $\Delta, C, \Delta', B, \Delta'' \vdash A$*.*

We shall also make use of the following usual fact on stability of provability under substitution of first-order terms.

Lemma 6.20 (Substitution)**.** *In* NJ_1 *and* NK_1*, if* $\Delta \vdash A$ *then* $\Delta[\mathsf{a}^\sigma/\mathsf{x}^\sigma] \vdash A[\mathsf{a}^\sigma/\mathsf{x}^\sigma]$*.*

6.2.2. Soundness and Completeness w.r.t. the Classical Semantics; Theories

Deduction in NK_1 (and thus in NJ_1) is sound w.r.t. the semantics of §6.1.3. Say that a sequent $A_1, \dots, A_n \vdash A$ is **valid** if the formula $(A_1 \times \cdots \times A_n) \Rightarrow A$ is valid in the sense of Def. 6.11.

Proposition 6.21. *If* $\Delta \vdash A$ *is derivable in* NK_1*, then* $\Delta \vdash A$ *is valid.*

Proposition 6.21 **fails** without the side conditions in the rules ($\forall$-I) and ($\exists$-E) of Fig. 1.

Example 6.22. *Replacing either of the rules* ($\forall$-I) *and* ($\exists$-E) *with*

$$\frac{\Delta \vdash A}{\Delta \vdash (\forall \mathsf{x}^\sigma)A} \qquad \textit{or} \qquad \frac{\Delta \vdash (\exists \mathsf{x}^\sigma)A \qquad \Delta, A \vdash B}{\Delta \vdash B}$$

would result in ***unsound*** *systems, since in each case we have* $(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)$*. Indeed, the wrong* $\forall$*-rule gives*

$$(\exists\text{-E})\ \frac{\dfrac{}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash (\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)} \qquad \dfrac{\dfrac{}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma),\ A(\mathsf{x}^\sigma) \vdash A(\mathsf{x}^\sigma)}}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma),\ A(\mathsf{x}^\sigma) \vdash (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)}}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)}$$

where the instance of ($\exists$-E) *respects the side condition* $\mathsf{x}^\sigma \notin \mathrm{FV}\big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma), (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big)$*, while the wrong* $\exists$*-rule gives*

$$(\forall\text{-I})\ \frac{\dfrac{\dfrac{}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash (\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)} \qquad \dfrac{}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma),\ A(\mathsf{x}^\sigma) \vdash A(\mathsf{x}^\sigma)}}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash A(\mathsf{x}^\sigma)}}{(\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma) \vdash (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)}$$

where the instance of ($\forall$-I) *respects the side condition* $\mathsf{x}^\sigma \notin \mathrm{FV}\big((\exists \mathsf{x}^\sigma)A(\mathsf{x}^\sigma), (\forall \mathsf{x}^\sigma)A(\mathsf{x}^\sigma)\big)$.

Similarly as with propositional logic, intuitionistic predicate logic is not complete w.r.t. the classical semantics. But it is indeed complete w.r.t. weaker semantics, see [SU06, §8.5] (see also [vD04, §5.3]).

On the other hand, **classical** predicate logic is complete w.r.t. the classical semantics. We recall here the relevant statements.

Definition 6.23. *Let* $\mathbf{\Phi}$ *be an* ***arbitrary*** *set of sentences and let* A *be a sentence.*

(1) We say that $\mathbf{\Phi}$ *is* ***valid*** *in* $\mathcal{M}$ *(or that* $\mathcal{M}$ *is a* ***model*** *of* $\mathbf{\Phi}$*), notation* $\mathcal{M} \models \mathbf{\Phi}$*, if* $\mathcal{M} \models A$ *for each* $A \in \mathbf{\Phi}$*.*

(2) We write $\mathbf{\Phi} \models A$ *if* $\mathcal{M} \models A$ *whenever* $\mathcal{M} \models \mathbf{\Phi}$*.*

(3) We write $\mathbf{\Phi} \vdash_{\mathsf{NK}_1} A$ *if there is a finite list* $\Delta \subseteq \mathbf{\Phi}$ *(i.e.* $A \in \mathbf{\Phi}$ *for each* $A \in \Delta$*) such that* $\Delta \vdash A$ *is derivable in* NK_1*.*

Note that $\mathbf{\Phi} \vdash A$ trivially implies $\mathbf{\Phi} \models A$.

Theorem 6.24 (Completeness [vD04, Thm. 3.1.3])**.** *If* $\mathbf{\Phi} \models A$ *then* $\mathbf{\Phi} \vdash_{\mathsf{NK}_1} A$*.*

Corollary 6.25 (Compactness)**.** *Fix a set of sentences* $\mathbf{\Phi}$*. If for every* ***finite*** $\mathbf{\Phi}_0 \subseteq \mathbf{\Phi}$ *there is a model* $\mathcal{M}$ *such that* $\mathcal{M} \models \mathbf{\Phi}_0$*, then there is a model* $\mathcal{M}$ *such that* $\mathcal{M} \models \mathbf{\Phi}$*.*

Proof. Exercise! □

We conclude with the notion of theory, which shall be of future use. Intuitively, a classical (resp. intuitionistic) theory is a set of sentences which is closed under classical (resp. intuitionistic) deduction.

Definition 6.26 (Theory)**.** *Let $\mathbf{\Phi}$ be a set of sentences.*

*(1) We say that $\mathbf{\Phi}$ is a **classical** theory if $A \in \mathbf{\Phi}$ whenever $\mathbf{\Phi} \vdash_{\mathsf{NK}_1} A$.*

*(2) We say that $\mathbf{\Phi}$ is an **intuitionistic** theory if $A \in \mathbf{\Phi}$ whenever $\mathbf{\Phi} \vdash_{\mathsf{NJ}_1} A$, where $\mathbf{\Phi} \vdash_{\mathsf{NJ}_1} A$ if there is a finite list $\Delta \subseteq \mathbf{\Phi}$ such that $\Delta \vdash A$ in NJ_1.*

*A theory $\mathbf{\Phi}$ is **consistent** if $\mathbf{\Phi} \nvdash_{\mathsf{ND}} \bot$ (where ND is NK_1 for a classical theory and ND is NJ_1 for an intuitionistic one).*

In both cases (classical or intuitionistic), every set of sentences $\mathbf{\Phi}$ is contained in a least (classical or intuitionistic) theory $\mathrm{th}(\mathbf{\Phi})$. Note that the Completeness Theorem 6.24 can be rephrased as:

- Every consistent classical theory has a model.

6.2.3. Negative Translations

The negative translations discussed in §2.5 extend to first-order predicate logic.

Definition 6.27 (Negative Translation)**.** *Extending Def. 2.37, the formula $A^\neg$ is defined by induction on A as follows:*

$$\begin{array}{rcl}
\mathsf{P}(\mathsf{a}_1^{\sigma_1}, \dots, \mathsf{a}_n^{\sigma_n})^\neg & := & \neg\neg\mathsf{P}(\mathsf{a}_1^{\sigma_1}, \dots, \mathsf{a}_n^{\sigma_n}) \\
\top^\neg & := & \top \\
\bot^\neg & := & \bot \\
(A \Rightarrow B)^\neg & := & A^\neg \Rightarrow B^\neg \\
(A \land B)^\neg & := & A^\neg \land B^\neg \\
(A \lor B)^\neg & := & \neg\neg(A^\neg \lor B^\neg) \\
\big((\forall \mathsf{x}^\sigma)A\big)^\neg & := & (\forall \mathsf{x}^\sigma)A^\neg \\
\big((\exists \mathsf{x}^\sigma)A\big)^\neg & := & \neg\neg(\exists \mathsf{x}^\sigma)A^\neg
\end{array}$$

The following are the respective extensions of Lem. 2.38 and Thm. 2.39.

Lemma 6.28. *For each formula A, NJ_1 proves $\neg\neg A^\neg \vdash A^\neg$.*

Theorem 6.29. *If $A_1, \dots, A_n \vdash A$ is derivable in NK_1, then $A_1^\neg, \dots, A_n^\neg \vdash A^\neg$ is derivable in NJ_1.*

We shall discuss Lem. 6.28 and Thm. 6.29 as (almost) direct instances of Prop. 7.67 in §7.3.4 below (on **Friedman's Translation** of classical to intuitionistic first-order arithmetic).

Corollary 6.30. *If $\vdash A$ is derivable in NK_1, then $\vdash A^\neg$ is derivable in NJ_1.*

Remark 6.31 (On Minimal Logic)**.** *Remark 2.41 extends to intuitionistic first-order logic, in the sense that if $\vdash A$ in NK_1 then $\vdash A^\neg$ can be derived in NJ_1 without* (ExFalso)*. It again suffices to observe that the statement of Lem. 6.28 actually holds in minimal logic and that $\bot \vdash A^\neg$ is provable in minimal logic for each formula A. We again refer to [TvD88a, Chap. 2, §3] for details.*

$$(\forall \mathsf{x})\,\big(\mathsf{x} =_{\mathrm{sk}\beta_0} \mathsf{x}\big) \qquad (\forall \mathsf{x}, \mathsf{y})\big(\mathsf{x} =_{\mathrm{sk}\beta_0} \mathsf{y} \;\Rightarrow\; \mathsf{y} =_{\mathrm{sk}\beta_0} \mathsf{x}\big)$$

$$(\forall \mathsf{x}, \mathsf{y}, \mathsf{z})\big(\mathsf{x} =_{\mathrm{sk}\beta_0} \mathsf{y} \;\Rightarrow\; \mathsf{y} =_{\mathrm{sk}\beta_0} \mathsf{z} \;\Rightarrow\; \mathsf{x} =_{\mathrm{sk}\beta_0} \mathsf{z}\big)$$

$$(\forall \mathsf{x}, \mathsf{x}', \mathsf{y}, \mathsf{y}')\big(\mathsf{x} =_{\mathrm{sk}\beta_0} \mathsf{x}' \;\Rightarrow\; \mathsf{y} =_{\mathrm{sk}\beta_0} \mathsf{y}' \;\Rightarrow\; \mathsf{x}\,\mathsf{y} =_{\mathrm{sk}\beta_0} \mathsf{x}'\,\mathsf{y}'\big)$$

$$(\forall \mathsf{x}, \mathsf{y})\big(\mathtt{k}\,\mathsf{x}\,\mathsf{y} =_{\mathrm{sk}\beta_0} \mathsf{x}\big) \qquad (\forall \mathsf{x}, \mathsf{y}, \mathsf{z})\big(\mathtt{s}\,\mathsf{x}\,\mathsf{y}\,\mathsf{z} =_{\mathrm{sk}\beta_0} \mathsf{x}\,\mathsf{z}\,(\mathsf{y}\,\mathsf{z})\big)$$

Figure 11: Closed Formulae for $\mathbf{\Gamma}_0$.

Remark 6.32 (On the Gödel-Gentzen and Kolmogorov Translations)**.** *Extending Rem. 2.43, the original Gödel-Gentzen translation (see e.g. [TvD88a, Chap. 2, Def. 3.4] or [vD04, Def. 5.2.7]) assumes*

$$\big((\exists \mathsf{x}^\sigma)A\big)^{\neg} \quad := \quad \neg(\forall \mathsf{x}^\sigma)\neg A^{\neg}$$

The Kolmogorov translation $(-)^{\neg\neg}$ (see e.g. [TvD88a, Chap. 2, 3.7]) extends with

$$\begin{array}{rcl} \mathsf{P}(\mathsf{a}_1^{\sigma_1},\dots,\mathsf{a}_n^{\sigma_n})^{\neg\neg} & := & \neg\neg\mathsf{P}(\mathsf{a}_1^{\sigma_1},\dots,\mathsf{a}_n^{\sigma_n}) \\ \big((\forall \mathsf{x}^\sigma)A\big)^{\neg\neg} & := & \neg\neg(\forall \mathsf{x}^\sigma)A^{\neg\neg} \\ \big((\exists \mathsf{x}^\sigma)A\big)^{\neg\neg} & := & \neg\neg(\exists \mathsf{x}^\sigma)A^{\neg\neg} \end{array}$$

6.2.4. Undecidability

A fundamental fact on (classical) first-order logic is the undecidability of provability (or equivalently of validity, by the Completeness Theorem 6.24, §6.2.2). This result, known as **Church's Theorem**, can be proved in many ways. See *e.g.* [BBJ07, §11] or [vD04, Cor. 7.7.12]. Note that Cor. 6.30 (§6.2.3) reduces the undecidability of provability in NJ_1 to the undecidability of provability in NK_1 ([SU06, Cor. 8.3.2]).

We note here that having at hand the (pure) untyped λ-calculus (actually Combinatory Logic, §3.7) allows for a quite direct proof of undecidability of NJ_1 and NK_1. See [Bar84, 7.4.11] for references on the following argument.

We consider the (one-sorted) finite first-order signature $\mathbf{\Sigma}(\mathcal{C}_0)$ whose closed terms are the closed terms $t \in \mathcal{C}_0$ of combinatory logic (§3.7), *i.e.* the terms over the grammar:

$$t, u \in \mathcal{C}_0 \quad ::= \quad t\ u \quad | \quad \mathtt{s} \quad | \quad \mathtt{k}$$

Remark 6.33. *Note that $\mathbf{\Sigma}(\mathcal{C}_0)$ actually has a binary function symbol for application (similarly as in §3.2.1).*

We also assume that $\mathbf{\Sigma}(\mathcal{C}_0)$ has one (infix) binary predicate symbol $(-) =_{\mathrm{sk}\beta_0} (-)$. Let $\mathbf{\Gamma}_0$ be the (**finite**) set of (closed) formulae consisting of the axioms of Fig. 3 (§3.7.1) together with the formulae of Fig. 11.

Proposition 6.34. *The following are equivalent for $t, u \in \mathcal{C}_0$:*

(i) $t =_{\mathrm{sk}\beta_0} u$ (in the sense of §3.7.1);

(ii) $\mathbf{\Gamma}_0 \vdash t =_{\mathrm{sk}\beta_0} u$ *in* NJ_1*;*

(iii) $\mathbf{\Gamma}_0 \vdash t =_{\mathrm{sk}\beta_0} u$ *in* NK_1*.*

Proof. First, if $t =_{\mathrm{sk}\beta_0} u$ in the sense of §3.7, then it is clear that $\mathbf{\Gamma}_0 \vdash t =_{\mathrm{sk}\beta_0} u$ is provable in NJ_1. Moreover, provability in NJ_1 trivially implies provability in NK_1.

Finally, note that $\mathcal{C}_0$ can be turned into a model of $\mathbf{\Sigma}(\mathcal{C}_0)$, in which the **predicate symbol** $(-) =_{\mathrm{sk}\beta_0} (-)$ of $\mathbf{\Sigma}(\mathcal{C}_0)$ is interpreted by the **relation** $=_{\mathrm{sk}\beta_0}$ of §3.7.1 (so that $\mathcal{C}_0 \models \mathbf{\Gamma}_0$). Assume now that $\mathbf{\Gamma}_0 \vdash t =_{\mathrm{sk}\beta_0} u$, where $t, u \in \mathcal{C}_0$. Note that t and u are closed terms over $\mathbf{\Sigma}(\mathcal{C}_0)$. Hence, it follows from the Soundness of NK_1 (Prop. 6.21, §6.2.2) that $\mathcal{C}_0 \models (t =_{\mathrm{sk}\beta_0} u)$, that is $t =_{\mathrm{sk}\beta_0} u$. □

Since the set $\mathbf{\Gamma}_0$ is finite and since $=_{\mathrm{sk}\beta_0}$ is undecidable (Cor. 3.65, §3.7.1), Prop. 6.34 gives the following.

Theorem 6.35 (Church)**.** *The following problems are undecidable:*

- *Given a finite first-order signature* $\mathbf{\Sigma}$ *and a first-order formula* A *over* $\mathbf{\Sigma}$*, decide whether* $\vdash A$ *in* NJ_1*.*
- *Given a finite first-order signature* $\mathbf{\Sigma}$ *and a first-order formula* A *over* $\mathbf{\Sigma}$*, decide whether* $\vdash A$ *in* NK_1*.*

Note that the above argument for Thm. 6.35 also gives the undecidability of the $(\forall, \Rightarrow)$-fragment of (minimal) first-order logic. See also [SU06, §8.8] for a proof of undecidability of NJ_1 (actually also of its $(\forall, \Rightarrow)$-fragment) relying on the normalization of proof-terms (§6.3).

6.2.5. The Failure of Glivenko's Theorem for Full First-Order Logic

As noted in Table 3, NJ_1 proves

$$\vdash \quad (\exists \mathsf{x}^\sigma)\neg\neg A(\mathsf{x}^\sigma) \quad \Rightarrow \quad \neg\neg(\exists \mathsf{x}^\sigma) A(\mathsf{x}^\sigma)$$

Recalling the proof of Prop. 2.47 (§2.6), it is easy to see that Glivenko's Theorem 2.48 extends to the following.

Theorem 6.36. *Let* A *be a first-order formula* ***with no universal quantifier*** $(\forall \mathsf{x}^\sigma)$*. Then* NK_1 *proves* $\vdash A$ *if and only if* NJ_1 *proves* $\vdash \neg\neg A$*.*

However, Thm. 6.36 **does not** extend to the full language of first-order logic. In particular:

Proposition 6.37. NJ_1 *proves* ***neither*** *of the following:*

(1) $\vdash \neg\neg(\forall \mathsf{x}_1^{\sigma_1} \dots \mathsf{x}_n^{\sigma_n})\big(\mathsf{P}(\mathsf{x}_1^{\sigma_1} \dots \mathsf{x}_n^{\sigma_n}) \vee \neg \mathsf{P}(\mathsf{x}_1^{\sigma_1} \dots \mathsf{x}_n^{\sigma_n})\big)$

(2) $\vdash (\forall \mathsf{x}^\sigma)\neg\neg A \Rightarrow \neg\neg(\forall \mathsf{x}^\sigma) A$

Proposition 6.37 has easy semantic proofs (outside the scope of this course), see [SU06, Prop. 8.5.3] (also [vD04, 5.3(f), p. 170] or [TvD88a, Ex. 5.9, Chap. 2]). Note that Prop. 6.37.(2) follows from Prop. 6.37.(1).

Remark 6.38. *The formula of Prop. 6.37.(2) is called the* ***Double Negation Shift*** (DNS). *It plays a crucial role in semi-constructive extensions of intuitionistic arithmetic, in relation to arithmetic formulations of weak versions of the Axiom of Choice (see e.g. [BBC98, BO05], also [Koh08, §11], outside the scope of this course).*

Remark 6.39. *We shall see in §7.3.3 (actually §8.3) that* NJ_1 *is* ***compatible*** *with* ***some*** *instances of*

$$(\neg\forall\text{-EM})\frac{}{\vdash \neg(\forall \mathsf{x}_1^{\sigma_1}\ldots\mathsf{x}_n^{\sigma_n})(A \vee \neg A)}$$

in the sense that NJ_1 *augmented with some instances of* $(\neg\forall\text{-EM})$ *is consistent (i.e. does not prove* $\vdash \bot$*). This can also be shown with semantic methods ([SU06, Prop. 8.5.3] or e.g. [TvD88a, Ex. 5.9, Chap. 2]), see also [TvD88a, Prop. 3.4, Chap. 4].*

6.2.6. Extraction

The distinctive properties of NJ_1 over NK_1 are the following. The proofs are deferred to §6.3.3. A comparison with NK_1 is sketched in §6.4.6.

Theorem 6.40. *In intuitionistic first-order predicate logic* (NJ_1)*,*

(1) $\vdash \bot$ *is not derivable;*

(2) from a proof of $\vdash A_1 \vee A_2$ *one can effectively compute an* $i \in \{1,2\}$ *and a proof of* $\vdash A_i$*;*

(3) $\vdash \mathsf{P}(\mathsf{x}_1,\ldots,\mathsf{x}_n) \vee \neg\mathsf{P}(\mathsf{x}_1,\ldots,\mathsf{x}_n)$ *is* ***not*** *derivable (where* $\mathsf{P} \in \mathrm{Pred}(\boldsymbol{\Sigma})(n)$*);*

(4) from a proof of $\vdash (\exists \mathsf{x}^\sigma)A$ *one can effectively compute a first-order term* a^σ *and a proof of* $\vdash A[\mathsf{a}^\sigma/\mathsf{x}^\sigma]$.

Theorem 6.40.(4) is often called the **Witness** (or **Existence**) **Property**. The following consequence of Thm 6.40.(4) is one of the main properties of NJ_1.

Corollary 6.41. *From a proof of* $\vdash (\forall \mathsf{x}_1^{\sigma_1}\ldots\mathsf{x}_n^{\sigma_n})(\exists \mathsf{y}^\tau)A$ *in* NJ_1 *(where* $n \geq 1$ *and* A *has free variables among* $\mathsf{x}_1^{\sigma_1},\ldots,\mathsf{x}_n^{\sigma_n},\mathsf{y}^\tau$*), one can effectively compute a first-order term* $\mathsf{a}^\tau = \mathsf{a}^\tau(\mathsf{x}_1^{\sigma_1},\ldots,\mathsf{x}_n^{\sigma_n})$ *and a proof of* $\vdash (\forall \mathsf{x}_1^{\sigma_1}\ldots\mathsf{x}_n^{\sigma_n})A[\mathsf{a}^\tau/\mathsf{y}^\tau]$ *in* NJ_1.

Proof. We omit sorts. From a proof of $\vdash (\forall \mathsf{x}_1\ldots\mathsf{x}_n)(\exists \mathsf{y})A$ in NJ_1 one gets a proof of $\vdash (\exists \mathsf{y})A$. Then Thm. 6.40.(4) provides a first-order term $\mathsf{a} = \mathsf{a}(\mathsf{x}_1,\ldots,\mathsf{x}_n)$ and a proof of $\vdash A[\mathsf{a}/\mathsf{y}]$ in NJ_1 (note that the variables of a possibly not in A can be substituted by variables among $\mathsf{x}_1,\ldots,\mathsf{x}_n$). One thus obtains a proof of $\vdash (\forall \mathsf{x}_1\ldots\mathsf{x}_n)A[\mathsf{a}/\mathsf{y}]$. □

The effective character of Thm. 6.40 and Cor. 6.41 should be contrasted with the undecidability of provability in NJ_1 (Thm. 6.35, §6.2.4).

We shall prove Thm. 6.40 with a normalizing λ-calculus of proof-terms in §6.3 below. Corollary 6.41 is [SU06, Cor. 8.3.4 & Cor. 8.3.5], a particular case of [Bus98a, Thm. 3.1.3] (see also [TS00, Thm. 4.2.3 & Thm. 4.2.4]), all obtained by normalization of Gentzen-style **sequent calculi**. Weaker properties can be obtained by semantic methods (see *e.g.* [vD04, Thm. 5.4.2 & Thm. 5.4.3]).

We come back on Cor. 6.41 in §6.4.6 below.

6.3. Proof-Terms for Intuitionistic First-Order Predicate Logic

We now turn to proof-terms for NJ_1. Besides [SU06, §8.7], standard references include [Gal95], [TvD88b, Chap. 10, §8] and [TS00, §2.2].

Consider a first-order signature $\mathbf{\Sigma}$. For simplicity we assume that $\mathbf{\Sigma}$ is one-sorted, but the following is easily extended to many-sorted signatures. We use the notational conventions of §6.1 for one-sorted signatures.

The proof-terms for NJ_0 (§4.4) can be adapted to yield proof-terms for NJ_1. The main ideas come from the Brouwer-Heyting-Kolmogorov (BHK) interpretation of intuitionistic first-order predicate logic. Recall from §4.1 that the BHK interpretation is an **informal** interpretation of intuitionistic **proofs**. The clauses given in §4.1 for propositional logic may be extended as follows for (intuitionistic) first-order logic:

- a "*proof*" of $(\forall \mathsf{x})A$ is a "*function*" which takes a term $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ to a "*proof*" of $A[\mathsf{a}/\mathsf{x}]$;
- a "*proof*" of $(\exists \mathsf{x})A$ is a (dependent) pair of a term $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ and a "*proof*" of $A[\mathsf{a}/\mathsf{x}]$.

where the term "*proof*" should be understood as some informal notion of "witness of evidence", and where the term "*function*" is not fully specified. We refer to [SU06, §8.2] and (again) to [TvD88a, Chap. 1, §3.1] for more on the BHK interpretation of intuitionistic first-order logic.

In view of the Curry-Howard correspondence for NJ_0, following Rem. 4.44 (§4.4.5) it makes sense to think that the word "*function*" above refers to some λ-abstraction. We shall extend the proof-terms of §4.4 with the following constructs:

- a λ-abstraction $\lambda \mathsf{x}.t$ where $\mathsf{x} \in \mathcal{V}$ is a **first-order** variable, and the corresponding elimination $t\mathsf{a}$ where $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ is a **first-order** term;
- a pairing $\langle \mathsf{a}, t \rangle$ where $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$, and the corresponding elimination $\mathtt{let}\ \langle \mathsf{x}, y \rangle = t\ \mathtt{in}\ u$, where $\mathsf{x} \in \mathcal{V}$ (see Fig. 12).

$$(\forall\text{-I})\ \frac{\mathcal{E} \vdash t : A}{\mathcal{E} \vdash \lambda \mathsf{x}.t : (\forall \mathsf{x})A}(\mathsf{x} \notin \mathrm{FV}(\mathcal{E})) \qquad (\forall\text{-E})\ \frac{\mathcal{E} \vdash t : (\forall \mathsf{x})A}{\mathcal{E} \vdash t\mathsf{a} : A[\mathsf{a}/\mathsf{x}]}$$

$$(\exists\text{-I})\ \frac{\mathcal{E} \vdash t : A[\mathsf{a}/\mathsf{x}]}{\mathcal{E} \vdash \langle \mathsf{a}, t\rangle : (\exists \mathsf{x})A} \qquad (\exists\text{-E})\ \frac{\mathcal{E} \vdash t : (\exists \mathsf{x})A \qquad \mathcal{E}, y : A \vdash u : B}{\mathcal{E} \vdash \mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u : B}\ (\mathsf{x} \notin \mathrm{FV}(\mathcal{E}) \cup \mathrm{FV}(B))$$

Figure 12: Proof-Terms for Intuitionistic First-Order Predicate Logic: Quantifier Rules.

Formally, the **proof-terms** for NJ_1 over $(\mathbf{\Sigma}, \mathcal{V})$ are given by the following grammar:

$$\begin{array}{rcl} t, u \in \Lambda(\mathbf{\Sigma}, \mathcal{V}) & ::= & x \mid \lambda x.t \mid tu \mid \langle t, u\rangle \mid \pi_1 t \mid \pi_2 t \\ & \mid & \mathtt{in}_1\, t \mid \mathtt{in}_2\, t \mid \langle\rangle \mid \mathtt{case}_\perp\ t\ \{\} \\ & \mid & \mathtt{case}\ t\ \{\mathtt{in}_1\, x_1 \mapsto u_1 \mid \mathtt{in}_2\, x_2 \mapsto u_2\} \\ & \mid & \lambda \mathsf{x}.t \mid t\,\mathsf{a} \\ & \mid & \langle \mathsf{a}, t\rangle \mid \mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u \end{array}$$

where $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ and $\mathsf{x} \in \mathcal{V}$.

Note that we thus have **two** universes of terms:

- the **first-order** terms $\mathsf{a}, \mathsf{b}, \ldots \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$, with variables $\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots \in \mathcal{V}$;
- the **proof-terms** $t, u, \ldots \in \Lambda(\mathbf{\Sigma}, \mathcal{V})$, with variables $x, y, z, \ldots \in \mathcal{X}$ (§3.2.1).

Warning 6.42 (Variable Binding). *The variable x is bound in $\lambda \mathsf{x}.t$. Moreover, the construction $\mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u$ binds x and y in u (but not in t). We thus assume the corresponding extension of α-conversion (§3.2.3 and §3.2.4).*

The notions of free-variable (Def. 3.3, §3.2.3) and of capture-avoiding substitution (Def. 3.8, §3.3) are adapted accordingly.

Concerning typing, we take as types the formulae of first-order predicate logic over $(\mathbf{\Sigma}, \mathcal{V})$, namely:

$$\begin{array}{rcl} A, B & ::= & \mathsf{P}(\mathsf{a}_1, \ldots, \mathsf{a}_n) \mid A \Rightarrow B \mid A \wedge B \mid A \vee B \mid \top \mid \bot \\ & \mid & (\forall \mathsf{x})A \mid (\exists \mathsf{x})A \end{array}$$

where $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(n)$ and $\mathsf{a}_i \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ for each $i = 1, \ldots, n$.

The typing rules are those of §4.4.5 augmented with the additional typing rules of Fig. 12, where given $\mathcal{E} = x_1 : A_1, \ldots, x_n : A_n$, we let $\mathrm{FV}(\mathcal{E})$ stand for $\mathrm{FV}(A_1) \cup \cdots \cup \mathrm{FV}(A_n)$ (not to be confused with $\mathrm{dom}(\mathcal{E}) = \{x_1, \ldots, x_n\}$).

Theorem 4.37 (§4.4.5) trivially extends, with the erasure of proof-terms from typing rules depicted in Fig. 13.

Theorem 6.43. *The sequent $A_1, \ldots, A_n \vdash A$ is derivable in NJ_1 if and only if there is a proof-term t such that $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$.*

$$\frac{x_1 : A_1, \dots, x_n : A_n, \vdash t : A}{x_1 : A_1, \dots, x_n : A_n \vdash \lambda \mathsf{x}.t : (\forall \mathsf{x})A} \quad \rightsquigarrow \quad \frac{A_1, \dots, A_n \vdash A}{A_1, \dots, A_n \vdash (\forall \mathsf{x})A} \qquad (\forall\text{-I})$$

$$\frac{x_1 : A_1, \dots, x_n : A_n, \vdash t : (\forall \mathsf{x})A}{x_1 : A_1, \dots, x_n : A_n \vdash t\mathsf{a} : A[\mathsf{a}/\mathsf{x}]} \quad \rightsquigarrow \quad \frac{A_1, \dots, A_n \vdash (\forall \mathsf{x})A}{A_1, \dots, A_n \vdash A[\mathsf{a}/\mathsf{x}]} \qquad (\forall\text{-E})$$

$$\frac{x_1 : A_1, \dots, x_n : A_n, \vdash t : A[\mathsf{a}/\mathsf{x}]}{x_1 : A_1, \dots, x_n : A_n \vdash \langle \mathsf{a}, t \rangle : (\exists \mathsf{x})A} \quad \rightsquigarrow \quad \frac{A_1, \dots, A_n \vdash A[\mathsf{a}/\mathsf{x}]}{A_1, \dots, A_n \vdash (\exists \mathsf{x})A} \qquad (\exists\text{-I})$$

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash t : (\exists \mathsf{x})A \qquad x_1 : A_1, \dots, x_n : A_n,\, y : A \vdash u : B}{x_1 : A_1, \dots, x_n : A_n \vdash \mathtt{let}\ \langle \mathsf{x}, y \rangle = t\ \mathtt{in}\ u : B} \qquad (\exists\text{-E})$$

$$\rightsquigarrow \quad \frac{A_1, \dots, A_n \vdash (\exists \mathsf{x})A \qquad A_1, \dots, A_n,\, A \vdash B}{A_1, \dots, A_n \vdash B}$$

Figure 13: Proof-Terms for NJ_1: Typing and Deduction Rules.

The other important properties of NJ_1 (essentially consequences of Thm. 6.43 together with suitable normalization results) are presented in §6.3.3 and further discussed in §6.4.6.

Example 6.44. *For instance* $\vdash \lambda h.\langle \mathsf{y}, h\mathsf{y} \rangle : (\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A$ *with typing derivation*

$$\cfrac{\cfrac{\cfrac{\cfrac{}{h : (\forall \mathsf{x})A \vdash h : (\forall \mathsf{x})A}}{h : (\forall \mathsf{x})A \vdash h\mathsf{y} : A[\mathsf{y}/\mathsf{x}]}}{h : (\forall \mathsf{x})A \vdash \langle \mathsf{y}, h\mathsf{y} \rangle : (\exists \mathsf{x})A}}{\vdash \lambda h.\langle \mathsf{y}, h\mathsf{y} \rangle : (\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A}$$

Exercise 6.45. *Give a proof term t such that* $\vdash t : \neg\neg(\forall \mathsf{x})A(\mathsf{x}) \Rightarrow (\forall \mathsf{x})\neg\neg A(\mathsf{x})$.

Remark 6.46 (Open Terms and Assumption-Free Proofs)**.** *Recall the proof-term* $\vdash \lambda h.\langle \mathsf{y}, h\mathsf{y} \rangle : (\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A$ *from Ex. 6.44. Note that* $\lambda h.\langle \mathsf{y}, h\mathsf{y} \rangle$ *is open (it contains* y *free), while the formulae* $(\forall \mathsf{x})A$ *and* $(\exists \mathsf{x})A$ *may be closed. If* $\mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ *contains some closed term (i.e. if* $\mathbf{\Sigma}$ *has at least one constant), then we might replace* y *with such a closed term, and obtain a closed proof-term for* $(\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A$.

However $(\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A$ *is provable and thus valid without any such assumption on* $\mathbf{\Sigma}$*. This reflects the fact that models of first-order logic are always assumed to be non-empty (see Def. 6.8, §6.1.3).*

Remark 6.47 (On the Side Condition of $(\exists$-E$)$)**.** *Note that in the rule*

$$(\exists\text{-E})\ \frac{\mathcal{E} \vdash t : (\exists \mathsf{x})A \qquad \mathcal{E}, y : A \vdash u : B}{\mathcal{E} \vdash \mathtt{let}\ \langle \mathsf{x}, y \rangle = t\ \mathtt{in}\ u : B}$$

the variable x *is required not to occur free in* $\mathcal{E}$ *nor* B*, but it may occur free in the proof-term* u*. For instance, the proof-term for the* NJ_1*-derivation*

$$(\exists\text{-E})\ \dfrac{\dfrac{}{(\exists \mathsf{x})A \vdash (\exists \mathsf{x})A} \qquad \dfrac{\dfrac{}{(\exists \mathsf{x})A, A \vdash A}}{(\exists \mathsf{x})A, A \vdash (\exists \mathsf{x})A}}{(\exists \mathsf{x})A \vdash (\exists \mathsf{x})A}\ (\mathsf{x} \notin \mathrm{FV}((\exists \mathsf{x})A))$$

is

$$(\exists\text{-E})\ \dfrac{\dfrac{}{h : (\exists \mathsf{x})A \vdash h : (\exists \mathsf{x})A} \qquad \dfrac{\dfrac{}{h : (\exists \mathsf{x})A, y : A \vdash y : A}}{h : (\exists \mathsf{x})A, y : A \vdash \langle \mathsf{x}, y\rangle : (\exists \mathsf{x})A}}{h : (\exists \mathsf{x})A \vdash \mathtt{let}\ \langle \mathsf{x}, y\rangle = h\ \mathtt{in}\ \langle \mathsf{x}, y\rangle : (\exists \mathsf{x})A}\ (\mathsf{x} \notin \mathrm{FV}((\exists \mathsf{x})A))$$

Remark 6.48. *The major difference with the case of propositional logic is that first-order terms and variables are incorporated into the world of proof-terms. But first-order terms and proof-terms are still distinct entities. For instance, first-order variables are not declared in typing contexts, which leads to the apparent "paradox" of Rem. 6.46.*

This one of the reasons why we refrain from speaking of "Curry-Howard correspondence" here. The second reason is that the λ-terms considered here have no other raison d'être *than being proof-terms for* NJ_1.

The formalism which is often considered to be more interesting for a Curry-Howard perspective on predicate logic is the setting of ***dependent types****. See [SU06, Chap. 13] or [Uni13, §1] (see also [Bar92, §5.4] for the representation of predicate logic with dependent types). In particular, with dependent types* ***all*** *variables are declared in typing contexts, and there are first-order signatures such that (the analogue of)* $(\forall \mathsf{x})A \Rightarrow (\exists \mathsf{x})A$ *is the type of no proof-term in the empty context.*

6.3.1. Structural Properties

We state some basic structural properties on proof-terms, proved by straightforward induction on typing derivations.

Lemma 6.49 (Structural Properties)**.**

(Weakening) *If* $\mathcal{E} \vdash t : A$ *and* $x \notin \mathrm{dom}(\mathcal{E})$ *then* $\mathcal{E}, x : B \vdash t : A$.

(Contraction) *If* $\mathcal{E}, x : B, y : B \vdash t : A$ *then* $\mathcal{E}, x : B \vdash t[x/y] : A$.

(Exchange) *If* $\mathcal{E}, x : B, \mathcal{E}', y : C, \mathcal{E}'' \vdash t : A$, *then* $\mathcal{E}, y : C, \mathcal{E}', x : B, \mathcal{E}'' \vdash t : A$.

There are two relevant substitution properties on proof terms.

Lemma 6.50 (Substitution for Proof-Terms)**.**

(1) If $\mathcal{E}, y : B \vdash t : A$ *and* $\mathcal{E} \vdash u : B$, *then* $\mathcal{E} \vdash t[u/y] : B$.

(2) If $\mathcal{E} \vdash t : A$ *then* $\mathcal{E}[\mathsf{a}/\mathsf{x}] \vdash t : A[\mathsf{a}/\mathsf{x}]$.

6.3.2. Beta-Reduction

We now discuss the extension of β-reduction to the proof-terms for NJ_1. Similarly as in §4.4, we shall derive reductions on λ-terms from expected syntactic transformations of proofs.

Universal Quantifications. Consider the following reduction of a $(\forall\text{-I})/(\forall\text{-E})$-redex in natural deduction

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi}}{\Delta \vdash A}}{\Delta \vdash (\forall \mathsf{x})A}\;(\mathsf{x} \notin \mathrm{FV}(\Delta))}{\Delta \vdash A[\mathsf{a}/\mathsf{x}]} \qquad \rhd \qquad \dfrac{\overset{\vdots}{\Pi[\mathsf{a}/\mathsf{x}]}}{\Delta \vdash A[\mathsf{a}/\mathsf{x}]}$$

where $\Pi[\mathsf{a}/\mathsf{x}]$ is obtained from the Substitution Lemma 6.20 (§6.2), since $\mathsf{x} \notin \mathrm{FV}(\Delta)$. This suggests to allow for the following β-reduction on proof-terms

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi}}{\mathcal{E} \vdash t : A}}{\mathcal{E} \vdash \lambda\mathsf{x}.t : (\forall \mathsf{x})A}\;(\mathsf{x} \notin \mathrm{FV}(\mathcal{E}))}{\mathcal{E} \vdash (\lambda\mathsf{x}.t)\mathsf{a} : A[\mathsf{a}/\mathsf{x}]} \qquad \rhd_\beta \qquad \dfrac{\overset{\vdots}{\Pi[\mathsf{a}/\mathsf{x}]}}{\mathcal{E} \vdash t[\mathsf{a}/\mathsf{x}] : A[\mathsf{a}/\mathsf{x}]}$$

where $\Pi[\mathsf{a}/\mathsf{x}]$ is obtained by the Substitution Lemma 6.50.(2) on proof-terms.

We shall just extend β-reduction on untyped λ-terms with

$$(\lambda\mathsf{x}.t)\mathsf{a} \quad \rhd_\beta \quad t[\mathsf{a}/\mathsf{x}]$$

Existential Quantifications. Consider now the case of a $(\exists\text{-I})/(\exists\text{-E})$ redex in natural deduction:

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi_1}}{\Delta \vdash A[\mathsf{a}/\mathsf{x}]}}{\Delta \vdash (\exists \mathsf{x})A} \qquad \dfrac{\overset{\vdots}{\Pi_2}}{\Delta, A \vdash B}}{\Delta \vdash B}\;(\mathsf{x} \notin \mathrm{FV}(\Delta, B)) \qquad \rhd \qquad \dfrac{\overset{\vdots}{\Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]}}{\Delta \vdash B}$$

The derivation $\Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]$ is obtained as follows. First, by the Substitution Lemma 6.20 (§6.2), since $\mathsf{x} \notin \mathrm{FV}(\Delta, B)$ we have a derivation

$$\dfrac{\overset{\vdots}{\Pi_2[\mathsf{a}/\mathsf{x}]}}{\Delta, A[\mathsf{a}/\mathsf{x}] \vdash B}$$

Then, $\Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]$ is obtained (similarly as in §4.3.1) by replacing in $\Pi_2[\mathsf{a}/\mathsf{x}]$ each (Ax) rule

$$\dfrac{}{\Delta' \vdash A[\mathsf{a}/\mathsf{x}]}$$

(where Δ is necessarily a prefix of Δ') by a derivation of $\Delta' \vdash A[\mathsf{a}/\mathsf{x}]$ obtained from Π_1 by weakenings (Lem. 6.49, §6.3.1).

This suggests to allow for the following β-reduction on proof-terms

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ \Pi_1\end{array}}{\mathcal{E} \vdash t : A[\mathsf{a}/\mathsf{x}]}}{\mathcal{E} \vdash \langle \mathsf{a}, t\rangle : (\exists \mathsf{x})A} \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_2\end{array}}{\mathcal{E}, y : A \vdash u : B}}{\mathcal{E} \vdash \mathtt{let}\ \langle \mathsf{x}, y\rangle = \langle \mathsf{a}, t\rangle\ \mathtt{in}\ u : B} \qquad \rhd_\beta \qquad \dfrac{\begin{array}{c}\vdots\\ \Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]\end{array}}{\mathcal{E} \vdash u[\mathsf{a}/\mathsf{x}][t/y] : B}$$

where $\mathsf{x} \notin \mathrm{FV}(\mathcal{E}) \cup \mathrm{FV}(B)$. The typing derivation $\Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]$ is obtained similarly as for the case of natural deduction (without proof terms) just described. First, since $\mathsf{x} \notin \mathrm{FV}(\mathcal{E}) \cup \mathrm{FV}(B)$ the Substitution Lemma 6.50.(2) on proof-terms gives a typing derivation

$$\dfrac{\begin{array}{c}\vdots\\ \Pi_2[\mathsf{a}/\mathsf{x}]\end{array}}{\mathcal{E}, y : A[\mathsf{a}/\mathsf{x}] \vdash u[\mathsf{a}/\mathsf{y}] : B}$$

We then conclude with Lem. 6.50.(1):

$$\dfrac{\begin{array}{c}\vdots\\ \Pi_2[\mathsf{a}/\mathsf{x}][\Pi_1/A[\mathsf{a}/\mathsf{x}]]\end{array}}{\mathcal{E} \vdash u[\mathsf{a}/\mathsf{x}][t/y] : B}$$

We shall therefore extend β-reduction on untyped λ-terms with

$$\mathtt{let}\ \langle \mathsf{x}, y\rangle = \langle \mathsf{a}, t\rangle\ \mathtt{in}\ u \quad \rhd_\beta \quad u[\mathsf{a}/\mathsf{x}][t/y]$$

6.3.3. The Full System and its Main Properties

Summarizing the discussion of §6.3.2, we consider on the proof-terms of NJ_1 the relation of β-reduction given by extending Fig. 5 (§4.4.3) with Fig. 14. More precisely, we first define the relation $\rhd_0$ with the basic rules of Fig. 14 (and Fig. 5), and then let $\rhd_\beta$ be the closure of $\rhd_0$ under the congruence rules of Fig. 14 (and Fig. 5).

The main properties on (typed) β-reduction of §4.4.4 straightforwardly extend to the proof-terms of NJ_1.

Proposition 6.51 (Subject Reduction). *If $\mathcal{E} \vdash t : T$ and $t \rhd_\beta u$ then $\mathcal{E} \vdash u : T$.*

Theorem 6.52 (Strong Normalization). *If $\mathcal{E} \vdash t : T$ then t is strongly β-normalizing.*

Theorem 6.52 is proven in §6.5 (Cor. 6.71), by reduction to the strong normalization of the simply-typed λ-calculus with sums and products (Thm. 4.31, §4.4.4, a.k.a. Thm. 5.32, §5.5.3).

Theorem 6.53 (Confluence). *For each typing context $\mathcal{E}$ and each type T, the relation $\rhd_\beta$ is confluent on $|\mathcal{E} \vdash T|$ (where $|\mathcal{E} \vdash T|$ is defined as in Rem. 4.11, §4.2.2).*

Basic Rules: extension of $\rhd_0$ (Fig. 5) with

$$(\lambda \mathsf{x}.t)\mathsf{a} \quad \rhd_0 \quad t[\mathsf{a}/\mathsf{x}] \qquad\qquad \mathtt{let}\ \langle \mathsf{x}, y\rangle = \langle \mathsf{a}, t\rangle\ \mathtt{in}\ u \quad \rhd_0 \quad u[\mathsf{a}/\mathsf{x}][t/y]$$

Congruence Rules: $\rhd_\beta$ is the least relation containing $\rhd_0$ and closed under the rules of Fig. 5 and

$$\frac{t \ \rhd_\beta\ t'}{t\,\mathsf{a} \ \rhd_\beta\ t'\,\mathsf{a}} \qquad\qquad \frac{t \ \rhd_\beta\ t'}{\lambda \mathsf{x}.t \ \rhd_\beta\ \lambda \mathsf{x}.t'} \qquad\qquad \frac{t \ \rhd_\beta\ t'}{\langle \mathsf{a}, t\rangle \ \rhd_\beta\ \langle \mathsf{a}, t'\rangle}$$

$$\frac{t \ \rhd_\beta\ t'}{\mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u \ \rhd_\beta\ \mathtt{let}\ \langle \mathsf{x}, y\rangle = t'\ \mathtt{in}\ u}$$

$$\frac{u \ \rhd_\beta\ u'}{\mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u \ \rhd_\beta\ \mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u'}$$

Figure 14: Extension of Fig. 5 (§4.4.3) for β-Reduction.

Lemma 6.54 (Normal Forms in the Empty Context)**.** *Let t be a (possibly open) proof-term typable in the empty context and in β-normal form. Then t is of one of the following forms:*

$$\lambda x.u \qquad \langle\rangle \qquad \langle u, v\rangle \qquad \mathtt{in}_1 u \qquad \mathtt{in}_2 u \qquad \lambda \mathsf{x}.u \qquad \langle \mathsf{a}, u\rangle$$

Proof. Similarly as for Lem. 4.33, the proof is by induction on t. We only discuss the new cases.

Case of $t = u\mathsf{a}$.

A direct inspection of the typing rules reveals that we must have

$$\frac{\vdash u : (\forall \mathsf{x})A}{\vdash u\mathsf{a} : A[\mathsf{a}/\mathsf{x}]}$$

where u is in β-normal form and typed in the empty context. It follows from the induction hypothesis that u must be of the form $\lambda \mathsf{x}.v$, but this is impossible since then $t = (\lambda \mathsf{x}.v)\mathsf{a}$ would not be in β-normal form.

Case of $t = (\mathtt{let}\ \langle \mathsf{x}, y\rangle = u\ \mathtt{in}\ v)$.

A direct inspection of the typing rules reveals that we must have

$$\frac{\vdash u : (\exists \mathsf{x})A \qquad y : A \vdash v : B}{\vdash \mathtt{let}\ \langle \mathsf{x}, y\rangle = u\ \mathtt{in}\ v : B}$$

where u is in β-normal form and typed in the empty context. It follows from the induction hypothesis that u must be of the form $\langle \mathsf{a}, u'\rangle$, but this is impossible since then t would not be in β-normal form. □

We now arrive at the main property of NJ_1, namely Thm. 6.40 (§6.2.6). Theorem. 6.40 is actually an immediate consequence of the following.

Theorem 6.55. *In intuitionistic first-order predicate logic (NJ_1),*

(1) there is no proof-term t such that $\vdash t : \bot$;

(2) if $\vdash t : A_1 \vee A_2$ then there is an $i \in \{1,2\}$ and a proof-term u_i such that $t \rhd^* \mathtt{in}_i\, u_i$ and $\vdash u_i : A_i$;

(3) there is no proof-term t such that $\vdash t : \mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n) \vee \neg\mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n)$ (where $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(n)$);

(4) if $\vdash t : (\exists \mathsf{x})A$ then there is some first-order term a and some proof-term u such that $t \rhd^*_\beta \langle \mathsf{a}, u\rangle$ and $\vdash u : A[\mathsf{a}/\mathsf{x}]$.

Proof. Items (1), (2) and (4) are direct consequences of Normalization (Thm. 6.52), Subject Reduction (Prop. 6.51) and Lem. 6.54.

Item (3) is proven similarly as for the analogous property for NJ_0 (Rem. 2.12, §2.1.1). Namely, if NJ_1 proved $\mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n) \vee \neg\mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n)$ then item (2) would give either a proof of $\mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n)$ or a proof of $\neg\mathsf{P}(\mathsf{x}_1, \dots, \mathsf{x}_n)$, contradicting the Soundness of NK_1 (Prop. 6.21, §6.2.2). $\square$

Remark 6.56. *Let us look at the shape of the proof-terms for Cor. 6.41 (§6.2.6).*

Let t such that $\vdash t : (\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$. By the results above, we can assume that t is of the form $\lambda \mathsf{x}_1 \dots \mathsf{x}_n.u$ with u in β-normal form. Moreover, since the x_i's are first-order variables, we must have $\vdash u : (\exists \mathsf{y})A$, so that (since u is normal) u is of the form $\langle \mathsf{a}, v\rangle$ with $\vdash v : A[\mathsf{a}/\mathsf{x}]$. It follows that Cor. 6.41 (§6.2.6) could have been obtained from the following:

- *A β-normal proof-term for $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$ is of the form $\vdash \lambda \mathsf{x}_1 \dots \mathsf{x}_n.\langle \mathsf{a}, v\rangle : (\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$ where $\vdash \lambda \mathsf{x}_1 \dots \mathsf{x}_n.v : (\forall \mathsf{x}_1 \dots \mathsf{x}_n)A[\mathsf{a}/\mathsf{y}]$.*

6.4. First-Order Predicate Logic with Equality

The extension of the above to logic with equality holds no surprise. We review the syntax and semantics of predicate logic with equality in §6.4.1–§6.4.5. Finally, in §6.4.6 we come back on the witness property of NJ_1 (Cor. 6.41, §6.2.6), and briefly compare it to classical logic.

6.4.1. The Language of First-Order Predicate Logic with Equality

Let $\mathbf{\Sigma}$ be a first-order signature and $\mathcal{V}$ be a collection of $\mathbf{\Sigma}$-sorted first-order variables. First-order predicate logic with equality over $(\mathbf{\Sigma}, \mathcal{V})$ simply extends first-order predicate logic over $(\mathbf{\Sigma}, \mathcal{V})$ with one (binary) **sorted** atomic equality predicate $(-) \doteq_\sigma (-)$ for

$$(\doteq\text{-I})\ \frac{}{\Delta \vdash \mathsf{a}^\sigma \doteq_\sigma \mathsf{a}^\sigma} \qquad\qquad (\doteq\text{-E})\ \frac{\Delta \vdash \mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma \qquad \Delta \vdash A[\mathsf{a}^\sigma/\mathsf{x}^\sigma]}{\Delta \vdash A[\mathsf{b}^\sigma/\mathsf{x}^\sigma]}$$

Figure 15: Equality Rules.

each sort $\sigma \in \mathrm{Sort}(\mathbf{\Sigma})$. Formally, the formulae of first-order predicate logic with equality over $(\mathbf{\Sigma}, \mathcal{V})$ are given by the grammar:

$$\begin{array}{rcl} A, B & ::= & (\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma) \quad | \quad \mathsf{P}(\mathsf{a}_1, \dots, \mathsf{a}_n) \\ & | & A \Rightarrow B \quad | \quad A \wedge B \quad | \quad A \vee B \quad | \quad \top \quad | \quad \bot \\ & | & (\forall \mathsf{x}^\sigma)A \quad | \quad (\exists \mathsf{x}^\sigma)A \end{array}$$

where $\mathsf{a}^\sigma, \mathsf{b}^\sigma \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})(\sigma)$ and $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\tau_1, \dots, \tau_n)$ with $\mathsf{a}_i \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})(\tau_i)$ for each $i = 1, \dots, n$.

Notation 6.57. *In the one-sorted case, we write $(-) \doteq (-)$ for the (unique) equality predicate.*

Remark 6.58. *Similarly as [vD04, §2.6], we have defined first-order **logic** with equality over **arbitrary** signatures $\mathbf{\Sigma}$, but we could as well (similarly as [Bus98a, §2]) have defined a notion of **signature with equality** (i.e. a signature $\mathbf{\Sigma}$ such that each* $\mathrm{Pred}(\mathbf{\Sigma})(\sigma, \sigma)$ *contains a distinguished predicate symbol $(-) \doteq_\sigma (-)$), and speak of **logic** with equality only when considering **signatures** with equality.*

Our choice stems from the fact that equality is (in general) assumed to satisfy some specific axioms (§6.4.2), which lead to specific interpretation in models (§6.4.3), as well as specific behaviour under negative translations (§6.4.4) and specific proof terms (§6.4.5).

The other notions of §6.1.2 extend straightforwardly to logic with equality:

$$\begin{array}{rcl} \mathrm{FV}(\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma) & := & \mathrm{FV}(\mathsf{a}^\sigma) \cup \mathrm{FV}(\mathsf{b}^\sigma) \\ (\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma)[\mathsf{c}^\tau/\mathsf{y}^\tau] & := & \big(\mathsf{a}^\sigma[\mathsf{c}^\tau/\mathsf{y}^\tau] \doteq_\sigma \mathsf{b}^\sigma[\mathsf{c}^\tau/\mathsf{y}^\tau]\big) \end{array}$$

6.4.2. Deduction for First-Order Predicate Logic with Equality

Natural deduction for intuitionistic (resp. classical) first-order predicate logic with equality is the extension of natural deduction intuitionistic (resp. classical) first-order predicate logic with the additional **equality rules** of Fig. 15.

Exercise 6.59. *Show that intuitionistic first-order predicate logic with equality proves the following:*

(1) $\Delta \vdash (\forall \mathsf{x}^\sigma)(\mathsf{x}^\sigma \doteq_\sigma \mathsf{x}^\sigma)$

(2) $\Delta \vdash (\forall \mathsf{x}^\sigma, \forall \mathsf{y}^\sigma)\big(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \;\Rightarrow\; \mathsf{y}^\sigma \doteq_\sigma \mathsf{x}^\sigma\big)$

(3) $\Delta \vdash (\forall \mathsf{x}^\sigma, \forall \mathsf{y}^\sigma, \forall \mathsf{z}^\sigma)\big(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \;\Rightarrow\; \mathsf{y}^\sigma \doteq_\sigma \mathsf{z}^\sigma \;\Rightarrow\; \mathsf{x}^\sigma \doteq_\sigma \mathsf{z}^\sigma\big)$

It is important to make it clear that the **rules** of Fig. 15 are equivalent to more usual **axioms** for equality. We refer to [vD04, §2.10] and [Bus98a, §2.2.1] for equivalent presentations of the same theories.

Exercise 6.60. *Show that over* NJ_1 *and* NK_1*, the equality rules of Fig. 15 are equivalent to the following equality* ***axioms****:*

$$\frac{}{\Delta \vdash (\forall \mathsf{x}^\sigma)(\mathsf{x}^\sigma \doteq_\sigma \mathsf{x}^\sigma)} \qquad \frac{}{\Delta \vdash (\forall \mathsf{x}^\sigma, \mathsf{y}^\sigma)\,(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \;\Rightarrow\; A[\mathsf{x}^\sigma/\mathsf{z}^\sigma] \;\Rightarrow\; A[\mathsf{y}^\sigma/\mathsf{z}^\sigma])}$$

Remark 6.61. *It is well-known (see e.g. [vD04, §2.10]) that the axiomatization of equality of Ex. 6.60 could as well have been presented in the (more refined but) equivalent following form*

$$\frac{}{\Delta \vdash (\forall \mathsf{x}^\sigma)\,(\mathsf{x}^\sigma \doteq_\sigma \mathsf{x}^\sigma)} \qquad \frac{}{\Delta \vdash (\forall \mathsf{x}^\sigma, \mathsf{y}^\sigma)\big(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \;\Rightarrow\; \mathsf{y}^\sigma \doteq_\sigma \mathsf{x}^\sigma\big)}$$

$$\frac{}{\Delta \vdash (\forall \mathsf{x}^\sigma, \mathsf{y}^\sigma, \mathsf{z}^\sigma)\big(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \;\Rightarrow\; \mathsf{y}^\sigma \doteq_\sigma \mathsf{z}^\sigma \;\Rightarrow\; \mathsf{x}^\sigma \doteq_\sigma \mathsf{z}^\sigma\big)}$$

$$\frac{\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(\sigma_1, \dots, \sigma_n; \tau)}{\Delta \vdash (\forall \mathsf{x}_1^{\sigma_1} \dots \mathsf{x}_n^{\sigma_n})(\forall \mathsf{y}_1^{\sigma_1} \dots \mathsf{y}_n^{\sigma_n})\left(\left(\bigwedge_{1 \le i \le n} \mathsf{x}_i^{\sigma_1} \doteq_{\sigma_i} \mathsf{y}_i^{\sigma_i}\right) \;\Rightarrow\; \mathsf{f}(\vec{\mathsf{x}}) \doteq_\tau \mathsf{f}(\vec{\mathsf{y}})\right)}$$

$$\frac{\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\sigma_1, \dots, \sigma_n)}{\Delta \vdash (\forall \mathsf{x}_1^{\sigma_1} \dots \mathsf{x}_n^{\sigma_n})(\forall \mathsf{y}_1^{\sigma_1} \dots \mathsf{y}_n^{\sigma_n})\left(\left(\bigwedge_{1 \le i \le n} \mathsf{x}_i^{\sigma_1} \doteq_{\sigma_i} \mathsf{y}_i^{\sigma_i}\right) \;\Rightarrow\; \mathsf{P}(\vec{\mathsf{x}}) \Leftrightarrow \mathsf{P}(\vec{\mathsf{y}})\right)}$$

6.4.3. Models of First-Order Predicate Logic with Equality

A **model** of first-order predicate logic **with equality** over a signature $\mathbf{\Sigma}$ is simply a model $\mathcal{M}$ of $\mathbf{\Sigma}$ in the sense of Def. 6.8 (§6.1.3). Satisfaction and validity (Def. 6.11) for formulae with equality assume that the atomic formulae $(-) \doteq_\sigma (-)$ are interpreted as plain equality over $\mathcal{M}(\sigma)$. Formally, the relation $\mathcal{M}, v \models A$ (Fig. 9) is extended with the clause

$$\mathcal{M}, v \models (\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma) \quad \text{iff} \quad [\![\mathsf{a}^\sigma]\!]v = [\![\mathsf{b}^\sigma]\!]v$$

Deduction in (intuitionistic and) classical first-order predicate logic with equality is sound w.r.t. the semantics of first-order predicate logic with equality.

Concerning completeness and compactness, (in the case of classical logic) Thm. 6.24 and Cor. 6.25 (§6.2.2) extend to the following, where $\mathbf{\Phi}, A$ consist of sentences with equality.

Theorem 6.62 (Completeness [vD04, Thm. 3.1.3])**.** *If* $\mathbf{\Phi} \models A$ *then* $\mathbf{\Phi} \vdash A$.

Corollary 6.63 (Compactness)**.** *Fix a set of sentences* $\mathbf{\Phi}$*. If for every* ***finite*** $\mathbf{\Phi}_0 \subseteq \mathbf{\Phi}$ *there is a model* $\mathcal{M}$ *such that* $\mathcal{M} \models \mathbf{\Phi}_0$*, then there is a model* $\mathcal{M}$ *such that* $\mathcal{M} \models \mathbf{\Phi}$.

$$(\doteq\text{-I})\ \frac{}{\mathcal{E} \vdash \mathsf{eq}_I\ \mathsf{a} : \mathsf{a} \doteq \mathsf{a}} \qquad\qquad (\doteq\text{-E})\ \frac{\mathcal{E} \vdash t : \mathsf{a} \doteq \mathsf{b} \qquad \mathcal{E} \vdash u : A[\mathsf{a}/\mathsf{x}]}{\mathcal{E} \vdash \mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u : A[\mathsf{b}/\mathsf{x}]}$$

Figure 16: Typing Rules for Equality.

Remark 6.64. *It is fairly standard to obtain Thm. 6.62 from Thm. 6.24 (completeness of first-order logic without equality, §6.2.2). The trick is the following. Given a first-order signature $\mathbf{\Sigma}$, let $\mathbf{\Sigma}(\doteq)$ be the extension of $\mathbf{\Sigma}$ with one additional predicate symbol $(-) \doteq_\sigma (-)$ in $\mathrm{Pred}(\mathbf{\Sigma}(\doteq))(\sigma,\sigma)$ for each sort $\sigma \in \mathrm{Sort}(\mathbf{\Sigma})$. Consider a model $\mathcal{M}$ of $\mathbf{\Sigma}(\doteq)$ in the sense of Def. 6.8 (§6.1.3), which (may not interpret $\doteq_\sigma$ as equality but) additionally validates all the following formulae:*

$$(\forall \mathsf{x}^\sigma)(\mathsf{x}^\sigma \doteq_\sigma \mathsf{x}^\sigma) \qquad\qquad (\forall \mathsf{x}^\sigma, \mathsf{y}^\sigma)\,(\mathsf{x}^\sigma \doteq_\sigma \mathsf{y}^\sigma \ \Rightarrow\ A[\mathsf{x}^\sigma/\mathsf{z}^\sigma] \ \Rightarrow\ A[\mathsf{y}^\sigma/\mathsf{z}^\sigma])$$

Consider in each $\mathcal{M}(\sigma)$ the relation

$$a \cong_\sigma b \quad \textit{iff} \quad \mathcal{M}, [a/\mathsf{x}^\sigma, b/\mathsf{y}^\sigma] \models (x^\sigma \doteq_\sigma y^\sigma)$$

It easily follows from the above axioms on $(\doteq_\sigma)_\sigma$ that $(\cong_\sigma)_\sigma$ is a family of equivalence relations on $(\mathcal{M}(\sigma))_\sigma$ which is moreover compatible with all function and predicate symbols of $\mathbf{\Sigma}$. Then by quotienting each $\mathcal{M}(\sigma)$ with $\cong_\sigma$ one obtains a model $\mathcal{M}/\!\cong$ such that for each sentence A (with equality) we have $\mathcal{M} \models A$ (in the sense of Def. 6.11, §6.1.3) iff $\mathcal{M}/\!\cong\ \models A$ (in the sense of first-order logic with equality). See e.g. [vD04, Proof of Lem. 3.1.11] for details.

6.4.4. Negative Translations for First-Order Predicate Logic with Equality

The negative translations mentioned in §6.2.3 can be straightforwardly adapted to equality. In particular, extend $(-)^\neg$ (Def. 6.27) with

$$(\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma)^\neg \quad := \quad \neg\neg\,(\mathsf{a}^\sigma \doteq_\sigma \mathsf{b}^\sigma)$$

Theorem 6.65. *If $\vdash A$ is derivable in NK_1 with equality, then $\vdash A^\neg$ is derivable in NJ_1 with equality.*

Theorem 6.65 is (an almost direct instance of) Cor. 7.69 (§7.3.4 below).

6.4.5. Proof-Terms for First-Order Predicate Logic with Equality

Similarly as in §6.3, we restrict to one-sorted signatures (and generalization to many-sorted signatures is straightforward).

A quite nice fact is that the equality rules of Fig. 15 (§6.4.2) admit decently behaved proof-terms. The typing rules are given in Fig. 16. They amount to extend the proof-terms for NJ_1 over $(\mathbf{\Sigma}, \mathcal{V})$ with

$$t, u \quad ::= \quad \dots \quad | \quad \mathsf{eq}_I\ \mathsf{a} \quad | \quad \mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u$$

where $\mathsf{a}, \mathsf{b} \in \mathrm{Ter}(\boldsymbol{\Sigma}, \mathcal{V})$.

Beta-reduction is defined from the following obvious proof-reduction:

$$\dfrac{\dfrac{}{\mathcal{E} \vdash \mathsf{eq}_I\ \mathsf{a} : \mathsf{a} \doteq \mathsf{a}} \qquad \dfrac{\Pi}{\mathcal{E} \vdash u : A[\mathsf{a}/\mathsf{x}]}}{\mathcal{E} \vdash \mathsf{eq}_E\ \mathsf{a}\ \mathsf{a}\ (\mathsf{eq}_I\ \mathsf{a})\ u : A[\mathsf{a}/\mathsf{x}]} \qquad \rhd_\beta \qquad \dfrac{\Pi}{\mathcal{E} \vdash u : A[\mathsf{a}/\mathsf{x}]}$$

Technically it is actually convenient to let $\rhd_\beta$ on **untyped** λ-terms be the contextual closure of the extension of $\rhd_0$ (Fig. 14, §6.3.3) with

$$\mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ (\mathsf{eq}_I\ \mathsf{c})\ u \quad \rhd_0 \quad u$$

Here, by "contextual closure", we mean closure under the congruence rules of Fig. 5 (§4.4.3), Fig. 14 (§6.3.3) and

$$\dfrac{t \ \rhd_\beta\ t'}{\mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u \ \rhd_\beta\ \mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t'\ u} \qquad \dfrac{u \ \rhd_\beta\ u'}{\mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u \ \rhd_\beta\ \mathsf{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u'}$$

All the statements of §6.3.3 remain true in NJ_1 with equality, and we do not restate them. Note however that one should add $(\mathsf{eq}_I\ \mathsf{a})$ among the β-normal forms in the empty context (Lem. 6.54). Normalization is discussed in §6.5.

6.4.6. On Term Extraction and Classical Logic

We come back on what is perhaps the most important property of NJ_1, namely witness extraction from NJ_1-proofs of $\forall\exists$-statements (Cor. 6.41, §6.2.6), and discuss some of its salient aspects w.r.t. classical logic. We first restate the result for logic with equality.

Corollary 6.66. *In NJ_1 with equality, from a proof of $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$ one can effectively compute a first-order term $\mathsf{a} = \mathsf{a}(\mathsf{x}_1, \dots, \mathsf{x}_n)$ and a proof of $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_n)A[\mathsf{a}/\mathsf{y}]$.*

Intuitionistic logic is crucial for Cor. 6.66. If NJ_1 proves $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$, then the latter is of course valid. But only assuming the validity of $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$ is not enough. In particular, given a valid closed formula $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A$ and a model $\mathcal{M}$, the Axiom of Choice gives a function $f : \mathcal{M}^n \to \mathcal{M}$ such that $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)A[f(\vec{\mathsf{x}})/\mathsf{y}]$ holds in $\mathcal{M}$. But there is no reason that $f : \mathcal{M}^n \to \mathcal{M}$ is the interpretation of some term $\mathsf{a} = \mathsf{a}(\mathsf{x}_1, \dots, \mathsf{x}_n)$ over the signature under consideration! Moreover, even if it were, there is no reason to get the **same** term a **for every** model $\mathcal{M}$.

Actually (essentially) the best that classical logic can say in this case is the following version of the remarkable **Herbrand's Theorem** ([Bus98a, 2.5.1], for which it is essential that equality can be axiomatized as in Rem. 6.61, §6.4.2).

Theorem 6.67 (Herbrand)**.** *Let $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A_0$ be a closed formula with equality, where A_0 is **quantifier-free** (i.e. A_0 contains no quantifier). In NK_1 with equality, from a proof of $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A_0$ one can effectively compute a number $k \geq 1$ and terms $\mathsf{a}_1, \dots, \mathsf{a}_k$ with variables among $\mathsf{x}_1, \dots, \mathsf{x}_n$ and such that*

$$\vdash \quad (\forall \mathsf{x}_1 \dots \mathsf{x}_n)\big(A_0[\mathsf{a}_1/\mathsf{y}] \vee \dots \vee A_0[\mathsf{a}_k/\mathsf{y}]\big)$$

$$
\begin{array}{rcl}
(\mathsf{a} \doteq \mathsf{b})^\circ & := & \mathtt{unit} \\
(\mathsf{P}(\mathsf{a}_1,\dots,\mathsf{a}_n))^\circ & := & o \\
(A \Rightarrow B)^\circ & := & A^\circ \to B^\circ \\
(A \wedge B)^\circ & := & A^\circ \times B^\circ \\
(A \vee B)^\circ & := & A^\circ + B^\circ \\
\top^\circ & := & \mathtt{unit} \\
\bot^\circ & := & \mathtt{void} \\
((\forall \mathsf{x})A)^\circ & := & \iota \to A^\circ \\
((\exists \mathsf{x})A)^\circ & := & \iota \times A^\circ
\end{array}
$$

Figure 17: The Translation $(-)^\circ$ of Formulae to Types.

We again insist on the effective character of Cor. 6.66 and Thm. 6.67, while provability in NJ_1 and NK_1 is undecidable (Thm. 6.35, §6.2.4). In particular, beware that the number $k \in \mathbb{N}$ in Thm. 6.67 cannot be computed from $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y})A_0$ alone ([Bus98a, 2.5.4], essentially because of the undecidability of first-order logic).

6.5. Normalization

In this §6.5, we prove the strong normalization of the (typed) proof-terms for NJ_1 **with equality**. As announced, we shall devise a translation $(-)^\circ$ from (typed) proof-terms to (typed) λ-terms with sums and products (§4.4.3) such that

$$t^\circ \quad \rhd_\beta^+ \quad u^\circ \qquad \text{whenever} \qquad t \quad \rhd_\beta \quad u$$

The strong normalization for NJ_1 with equality will then be inferred from the strong normalization of the simply-typed λ-calculus with sums and products (Thm. 4.31, §4.4.4, a.k.a. Thm. 5.32, §5.5.3). Different proofs can be found in *e.g.* [SU06, §8.7], [TvD88b, Chap. 10, §8.4] or [Gal95].

The key idea in the definition of $(-)^\circ$ is to represent first-order terms by usual λ-terms, and to represent first-order formulae by simple types (with sums and products). Hence the translation $(-)^\circ$ depends on the first-order signature under consideration.

Fix a first-order signature $\mathbf{\Sigma}$ and a set of first-order variables $\mathcal{V}$ that (as for the proof-terms of §6.3 and §6.4.5) we assume to be one-sorted.

The translation $(-)^\circ$ targets simply-typed λ-terms with types over the grammar

$$T, U \quad ::= \quad \iota \quad | \quad o \quad | \quad U \to T \quad | \quad T \times U \quad | \quad T + U \quad | \quad \mathtt{unit} \quad | \quad \mathtt{void}$$

where

- ι is a base type, which shall be used to type the λ-terms representing first-order terms over $(\mathbf{\Sigma}, \mathcal{V})$; and
- o is a base type, which shall be used to represent the $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})$.

$$
\begin{array}{rcll}
x^\circ & := & x & \\
(\lambda x.t)^\circ & := & \lambda x.t^\circ & \\
(tu)^\circ & := & t^\circ u^\circ & \\
\langle t, u\rangle^\circ & := & \langle t^\circ, u^\circ\rangle & \\
(\pi_1 t)^\circ & := & \pi_1 t^\circ & \\
(\pi_2 t)^\circ & := & \pi_2 t^\circ & \\
(\mathtt{in}_1\, t)^\circ & := & \mathtt{in}_1\, t^\circ & \\
(\mathtt{in}_2\, t)^\circ & := & \mathtt{in}_2\, t^\circ & \\
\big(\mathtt{case}\ t\ \{\mathtt{in}_1\, x_1 \mapsto u_1 \mid \mathtt{in}_2\, x_2 \mapsto u_2\}\big)^\circ & := & \mathtt{case}\ t^\circ\ \{\mathtt{in}_1\, x_1 \mapsto u_1^\circ \mid \mathtt{in}_2\, x_2 \mapsto u_2^\circ\} & \\
(\mathtt{case}_\bot\ t\ \{\})^\circ & := & \mathtt{case}_\bot\ t^\circ\ \{\} & \\
(\lambda \mathsf{x}.t)^\circ & := & \lambda \mathsf{x}^\circ.t^\circ & \\
(t\,\mathsf{a})^\circ & := & t^\circ\,\mathsf{a}^\circ & \\
\langle \mathsf{a}, t\rangle^\circ & := & \langle \mathsf{a}^\circ, t^\circ\rangle & \\
\big(\mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u\big)^\circ & := & \big(\lambda p.u^\circ[\pi_1 p/\mathsf{x}^\circ][\pi_2 p/y]\big)t^\circ & (p \notin \mathrm{FV}(u^\circ)) \\
(\mathtt{eq}_I\ \mathsf{a})^\circ & := & \langle\rangle & \\
(\mathtt{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u)^\circ & := & (\lambda x.\lambda y.y)\ t^\circ\ u^\circ &
\end{array}
$$

Figure 18: The Translation $(-)^\circ$ on Untyped Proof-Terms.

Figure 17 defines the translation $(-)^\circ$ of **first-order formulae** of NJ_1 with equality to **simple types** over the above grammar. Note that A° contains no first-order term, so that in particular $(A[\mathsf{a}/\mathsf{x}])^\circ = A^\circ$.

We now turn to terms. We assume given

- for each first-order variable $\mathsf{x} \in \mathcal{V}$, a variable $\mathsf{x}^\circ \in \mathcal{X}$;
- for each $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(n)$, a variable $\mathsf{f}^\circ \in \mathcal{X}$ together with a type

$$T_\mathsf{f} \quad := \quad \underbrace{\iota \to \cdots \to \iota}_{n \text{ times}} \to \iota$$

It then follows that for each first-order term $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ we obtain a typed λ-term

$$\mathcal{I} \quad \vdash \quad \mathsf{a}^\circ \quad : \quad \iota$$

where $\mathcal{I}$ is a **first-order** context, *i.e.* a typing context which consists only of declarations of the form $\mathsf{x}^\circ : \iota$ (where $\mathsf{x} \in \mathcal{V}$) or $\mathsf{f}^\circ : T_\mathsf{f}$ (where $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})$). The λ-term a° is defined by induction on $\mathsf{a} \in \mathrm{Ter}(\mathbf{\Sigma}, \mathcal{V})$ as

$$\mathsf{x}^\circ \quad := \quad \mathsf{x}^\circ \qquad \text{and} \qquad (\mathsf{f}(\mathsf{a}_1, \dots, \mathsf{a}_n))^\circ \quad := \quad \mathsf{f}^\circ\, \mathsf{a}_1^\circ \cdots \mathsf{a}_n^\circ$$

For the translation of proof-terms, we proceed in two steps. We first define a translation $(-)^\circ$ on **untyped** proof-terms in Fig. 18. Second, an easy induction on derivations shows that the translation $(-)^\circ$ preserves typing in the following sense, where given $\mathcal{E} = x_1 : A_1, \dots, x_n : A_n$, we write $\mathcal{E}^\circ$ for $x_1 : A_1^\circ, \dots, x_n : A_n^\circ$.

Lemma 6.68. *If $\mathcal{E} \vdash t : A$ in NJ_1 with equality, then there is a first-order context $\mathcal{I}$ such that $\mathcal{E}^\circ, \mathcal{I} \vdash t^\circ : A^\circ$.*

Proof. Exercise! □

We shall now show that the translation $(-)^\circ$ preserves β-reduction, in the sense that

$$t^\circ \quad \rhd^+_\beta \quad u^\circ \qquad \text{whenever} \qquad t \quad \rhd_\beta \quad u$$

Lemma 6.69 (Substitution)**.**

(1) $\big(t[\mathsf{b}/\mathsf{y}]\big)^\circ = t^\circ[\mathsf{b}^\circ/\mathsf{y}^\circ]$

(2) $\big(t[v/y]\big)^\circ = t^\circ[v^\circ/y]$

Proof. Exercise! □

Proposition 6.70. *If $t \rhd_\beta u$ then $t^\circ \rhd^+_\beta u^\circ$.*

Proof. Exercise! □

We then obtain the strong normalization of the proof-terms of NJ_1 with equality from the strong normalization of the simply-typed λ-calculus with sums and products (Thm. 4.31, §4.4.4, a.k.a. Thm. 5.32, §5.5.3).

Corollary 6.71 (Strong Normalization)**.** *If $\mathcal{E} \vdash t : A$ in NJ_1 with equality, then t is strongly normalizing.*

7. First-Order Arithmetic

First-order classical ("Peano") and intuitionistic ("Heyting") arithmetic (denoted resp. PA and HA) are fundamental systems, for reasons related to **Gödel's Incompleteness Theorems**.

The key fact is that primitive recursion (and in particular finite sequences) can be represented in first-order arithmetic, which makes the latter a reference setting w.r.t. its provable statements on recursive functions and finitary data-structures.

Moreover, the very representation of primitive recursion allows for the usual Gödel's coding of syntax, to the effect that proof-trees for first-order arithmetic can be represented within first-order arithmetic. The (usual) crucial consequence is that if first-order arithmetic is consistent, then it cannot prove its own consistency. This last fact puts first-order arithmetic at the basis of a hierarchy of systems (first-order, second-order and higher-order arithmetics, type theories, set theories, and fragments thereof)[16] which (assuming their consistency) can be compared w.r.t. their **proof-theoretic strength**, *i.e.* w.r.t. their ability to prove the consistency of weaker systems, as well as w.r.t.

[16]First-order arithmetic as considered in this course is actually not the weakest system which can be taken as such a basis (see *e.g.* [Bus98b] for an introduction to **weak** systems of arithmetic).

$$(\forall \mathsf{x}^\iota)\neg(\mathsf{Sx} \doteq_\iota 0) \qquad (\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)\big(\mathsf{Sx} \doteq_\iota \mathsf{Sy} \;\Rightarrow\; \mathsf{x} \doteq_\iota \mathsf{y}\big)$$

Figure 19: Basic Axioms on 0 and S.

$$(\forall \mathsf{x})\,(\mathsf{x} \doteq_\iota \mathsf{x}) \qquad (\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)\big(\mathsf{x} \doteq_\iota \mathsf{y} \;\Rightarrow\; \mathsf{y} \doteq_\iota \mathsf{x}\big)$$

$$(\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)(\forall \mathsf{z}^\iota)\big(\mathsf{x} \doteq_\iota \mathsf{y} \;\Rightarrow\; \mathsf{y} \doteq_\iota \mathsf{z} \;\Rightarrow\; \mathsf{x} \doteq_\iota \mathsf{z}\big)$$

$$(\forall \mathsf{x}_1^\iota \dots \mathsf{x}_n^\iota)(\forall \mathsf{y}_1^\iota \dots \mathsf{y}_n^\iota)\left(\left(\bigwedge_{1\le i\le n} \mathsf{x}_i \doteq_\iota \mathsf{y}_i\right) \;\Rightarrow\; \mathsf{f}(\vec{\mathsf{x}}) \doteq_\iota \mathsf{f}(\vec{\mathsf{y}})\right)$$

$$(\forall \mathsf{x}_1^\iota \dots \mathsf{x}_n^\iota)(\forall \mathsf{y}_1^\iota \dots \mathsf{y}_n^\iota)\left(\left(\bigwedge_{1\le i\le n} \mathsf{x}_i \doteq_\iota \mathsf{y}_i\right) \;\Rightarrow\; \mathsf{P}(\vec{\mathsf{x}}) \Leftrightarrow \mathsf{P}(\vec{\mathsf{y}})\right)$$

Figure 20: Equality Axioms.

which partial recursive functions they are able to prove to be total (references on this are [GLT89, TS00, Bee85]; see also [Sim10] for a different approach).

First-order arithmetics come under different presentations in the literature. For this reason, we begin in §7.1 with some key facts which are to be waited from virtually any **intuitionistic** first-order arithmetic.

We then discuss in §7.2 a presentation of classical first-order **Peano Arithmetic** (PA) whose language contains one function symbol for each primitive recursive function.

The corresponding intuitionistic system, called **Heyting Arithmetic** (HA) is then discussed in §7.3, with a particular emphasis on its relation to PA and on its extraction results. Of particular importance is Kreisel's Theorem 7.54 (§7.3.3), according to which HA and PA prove the same "Π^0_2-sentences" (*i.e.* $\forall\exists$-sentences over primitive recursive predicates), so that HA and PA prove the termination of the same algorithms.

In contrast with first-order logic (§6) we shall not look for a proper λ-calculus of proof-terms for HA. But there is an extension of the simply-typed λ-calculus associated to HA via a variant of the Curry-Howard correspondence. This typed λ-calculus, called **Gödel's System T**, shall be discussed separately in §8.

7.1. Theories of Natural Numbers with Induction

Recall from Def. 6.26 (§6.2.2) that a classical (resp. intuitionistic) theory is a set $\mathbf{\Phi}$ of sentences (*i.e.* closed formulae) of first-order predicate logic which is stable under classical (resp. intuitionistic) deduction.

In this §7 we shall be concerned with theories which contain at least the following.

Definition 7.1.

(1) We say that a first-order signature $\mathbf{\Sigma}$ is a ***signature with natural numbers*** *if there is a sort $\iota \in \mathsf{Sort}(\mathbf{\Sigma})$ such that $\mathbf{\Sigma}$ contains the constant symbol $0 : \iota$ and the function symbol $\mathsf{S} : \iota \to \iota$.*

(2) By a (classical resp. intuitionistic) ***theory of natural numbers*** *we mean a (classical resp. intuitionistic) theory* TH *on a signature* $\mathbf{\Sigma}$ *with natural numbers such that* TH *proves all the formulae of Fig. 19 and of Fig. 20 (where* $\mathsf{f} \in \mathrm{Fun}(\mathbf{\Sigma})(\iota, \dots, \iota; \iota)$ *and* $\mathsf{P} \in \mathrm{Pred}(\mathbf{\Sigma})(\iota, \dots, \iota)$*).*

(3) We say that a (classical resp. intuitionistic) theory of natural numbers TH *has* ***induction*** *if* TH *proves all universal closures of the following, where* A *contains no disjunction* ($\vee$)*:*

$$A[0/\mathsf{x}] \;\Rightarrow\; (\forall \mathsf{x}^\iota)\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big) \;\Rightarrow\; (\forall \mathsf{x}^\iota)A$$

Notation 7.2. *The constant symbol* $\mathsf{0}$ *is intended to represent the natural number* 0*, while the function symbol* $\mathsf{S}(-)$ *is intended to represent the successor function* $n \in \mathbb{N} \mapsto (n+1) \in \mathbb{N}$*.*

(1) We often write Sa *instead of* $\mathsf{S}(\mathsf{a})$ *(as in Fig. 19).*

(2) Given a natural number $n \in \mathbb{N}$*, the* ***numeral*** $\underline{n}$ *is the (closed) first-order term defined by induction on* n *as* $\underline{0} := \mathsf{0}$ *and* $\underline{n+1} := \mathsf{S}\underline{n}$*.*

Remark 7.3. *The equality axioms of Fig. 20 are reminiscent from Rem. 6.61 (§6.4.2). It is well-known (see e.g. [vD04, §2.10]) that these axioms are equivalent to the universal closures of*

$$\mathsf{a} \doteq_\iota \mathsf{a} \qquad \textit{and} \qquad \mathsf{a} \doteq_\iota \mathsf{b} \;\Rightarrow\; A[\mathsf{a}/\mathsf{x}^\iota] \;\Rightarrow\; A[\mathsf{b}/\mathsf{x}^\iota]$$

What may seem strange in Def. 7.1 is the condition that A contains no disjunction in the **induction axiom**

$$A[0/\mathsf{x}] \;\Rightarrow\; (\forall \mathsf{x}^\iota)\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big) \;\Rightarrow\; (\forall \mathsf{x}^\iota)A$$

In the classical case, this is obviously not a restriction since disjunctions $A \vee B$ can be defined as $\neg(\neg A \wedge \neg B)$. In the intuitionistic case, this is due to the following well-known fact (see *e.g.* [TvD88a, Chap. 3, §3.2], also [SU06, Prop. 9.5.1]).

Definition 7.4. *Assuming a signature with natural numbers, let*

$$A \mathbin{\dot\vee} B \quad := \quad (\exists \mathsf{z}^\iota)\left[\big((\mathsf{z} \doteq_\iota \mathsf{0}) \Rightarrow A\big) \;\wedge\; \big(\neg(\mathsf{z} \doteq_\iota \mathsf{0}) \Rightarrow B\big)\right]$$

Proposition 7.5. *Let* TH *be an intuitionistic theory of natural numbers. Then* TH *proves the following:*

(1) $A \Rightarrow A \mathbin{\dot\vee} B$

(2) $B \Rightarrow A \mathbin{\dot\vee} B$

If furthermore TH *has induction, then* TH *proves*

(3) $(A \mathbin{\dot\vee} B) \;\Rightarrow\; (A \Rightarrow C) \;\Rightarrow\; (B \Rightarrow C) \;\Rightarrow\; C$

(4) $(A \dot{\vee} B) \;\Leftrightarrow\; (A \vee B)$

Proof. Exercise! □

We finally arrive at the fundamental fact that intuitionistic arithmetics prove the **decidability of equality**, namely:

$$(\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)\big(\mathsf{x} \doteq_\iota \mathsf{y} \;\vee\; \neg(\mathsf{x} \doteq_\iota \mathsf{y})\big)$$

This fact is at the basis of Kreisel's Theorem 7.54 (§7.3.3) and of considerable further developments (see *e.g.* [TvD88a, TvD88b, Koh08]).

Proposition 7.6. *Let* TH *be an intuitionistic theory of natural numbers. Then* TH *proves the following:*

(1) $(\forall \mathsf{x}^\iota)\big(\mathsf{x} \doteq_\iota 0 \;\dot{\vee}\; \neg(\mathsf{x} \doteq_\iota 0)\big)$

If furthermore TH *has induction, then* TH *proves*

(2) $(\forall \mathsf{x}^\iota)\;\big(\mathsf{x} \doteq_\iota 0 \;\dot{\vee}\; (\exists \mathsf{y}^\iota)(\mathsf{x} \doteq_\iota \mathsf{S}\mathsf{y})\big)$

(3) $(\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)\big(\mathsf{x} \doteq_\iota \mathsf{y} \;\dot{\vee}\; \neg(\mathsf{x} \doteq_\iota \mathsf{y})\big)$

Proof. Exercise! □

Corollary 7.7. *Intuitionistic theories of natural numbers with induction prove*

(1) $(\forall \mathsf{x}^\iota)\big(\mathsf{x} \doteq_\iota 0 \;\vee\; \neg(\mathsf{x} \doteq_\iota 0)\big)$

(2) $(\forall \mathsf{x}^\iota)\;\big(\mathsf{x} \doteq_\iota 0 \;\vee\; (\exists \mathsf{y}^\iota)(\mathsf{x} \doteq_\iota \mathsf{S}\mathsf{y})\big)$

(3) $(\forall \mathsf{x}^\iota)(\forall \mathsf{y}^\iota)\big(\mathsf{x} \doteq_\iota \mathsf{y} \;\vee\; \neg(\mathsf{x} \doteq_\iota \mathsf{y})\big)$

7.2. Peano Arithmetic

Peano Arithmetic is usually defined as the (classical) first-order theory (denoted PA_0 in the following) of natural numbers with induction (§7.1) and with function symbols $(-)+(-)$, $(-)\times(-)$ for resp. addition and multiplication on natural numbers, together with their defining equations.

However, it is well-known (and it follows from results essentially due to Gödel) that PA_0 can be conservatively extended with function symbols for primitive recursive functions. This results in a system (denoted PA in the following) whose intuitionistic counterpart, called **Heyting Arithmetic** (denoted HA and to be defined in §7.3), is often considered to be the good version of intuitionistic first-order arithmetic (besides [SU06, §9.5], see *e.g.* [TvD88a, Chap. 3, §3], [Tro73, §1.3] and [Koh08, §3.2]).

We shall thus begin in §7.2.1 with some reminder on primitive recursive functions, and in particular on the representation of partial recursive functions (§7.2.2). The language and axioms of first-order arithmetic are presented in §7.2.3 and §7.2.4 respectively. Our version of Peano Arithmetic (PA) is then defined in §7.2.5, while §7.2.6 is a reminder on the main results toward **Gödel's Incompleteness Theorems**, which also introduces the concept of **provably total function**, an important notion for comparing the proof-theoretic strength (*i.e.* provability) of theories containing arithmetic, in particular in relation to normalization of typed λ-calculi (see *e.g.* [GLT89, TS00, Bee85]).

This §7.2 is mostly based on [SU06, §9], but we also refer to [vD04, BBJ07] for material on primitive recursive functions and on Gödel's Incompleteness Theorems. We also refer to [Odi99] for the modicum of recursion theory we shall need.

7.2.1. Primitive Recursive Functions

We recall the definition of the primitive recursive functions. See [vD04, §7.1], [TvD88a, Chap. 3, §1] or [SU06, §A3] (also [BBJ07, §6.1]).

Notation 7.8. *We use the following notations.*

(1) If f is a function from $\mathbb{N}^k$ to $\mathbb{N}$ ($k \geq 0$), then we write $f : \mathbb{N}^k \to \mathbb{N}$.

*(2) **Composition:** Given $f : \mathbb{N}^m \to \mathbb{N}$ and, $g_i : \mathbb{N}^k \to \mathbb{N}$ ($1 \leq i \leq m$), then define $f \circ \langle g_1, \dots, g_m\rangle : \mathbb{N}^k \to \mathbb{N}$ as*

$$(f \circ \langle g_1, \dots, g_m\rangle)(n_1, \dots, n_k) \quad := \quad f(g_1(n_1, \dots, n_k), \dots, g_m(n_1, \dots, n_k))$$

*(3) **Primitive Recursion:** Given $f : \mathbb{N}^k \to \mathbb{N}$ and $g : \mathbb{N}^{k+2} \to \mathbb{N}$, we write $\mathrm{Iter}\langle f, g\rangle : \mathbb{N}^{k+1} \to \mathbb{N}$ for the function defined as*

$$\begin{array}{rcl} \mathrm{Iter}\langle f, g\rangle(n_1, \dots, n_k, 0) & := & f(n_1, \dots, n_k) \\ \mathrm{Iter}\langle f, g\rangle(n_1, \dots, n_k, n+1) & := & g(n_1, \dots, n_k, n, \mathrm{Iter}\langle f, g\rangle(n_1, \dots, n_k, n)) \end{array}$$

*(4) The **Zero** function $\mathrm{Zero}^k : \mathbb{N}^k \to \mathbb{N}$ is*

$$\mathrm{Zero}^k(n_1, \dots, n_k) \quad := \quad 0$$

*(5) The **Successor** function $\mathrm{Succ} : \mathbb{N} \to \mathbb{N}$ is*

$$\mathrm{Succ}(n) \quad := \quad n+1$$

*(6) If $1 \leq i \leq k$, the **Projection** function $\pi^k_i : \mathbb{N}^k \to \mathbb{N}$ is*

$$\pi^k_i(n_1, \dots, n_k) \quad := \quad n_i$$

Definition 7.9 (Primitive Recursive Function)**.** *The set* PR *of **primitive recursive functions** is the smallest subset of $\bigcup_{k\in\mathbb{N}} \mathbb{N}^{\mathbb{N}^k}$ which is closed under the following rules:*

$$\frac{}{\mathrm{Zero}^k \in \mathrm{PR}(k)} \qquad \frac{}{\mathrm{Succ} \in \mathrm{PR}(1)} \qquad \frac{}{\pi^k_i \in \mathrm{PR}(k)}$$

$$\frac{f \in \mathrm{PR}(m) \qquad g_1 \in \mathrm{PR}(k) \quad \dots \quad g_m \in \mathrm{PR}(k)}{f \circ \langle g_1, \dots, g_m\rangle \in \mathrm{PR}(k)} \qquad \frac{f \in \mathrm{PR}(k) \qquad g \in \mathrm{PR}(k+2)}{\mathrm{Iter}\langle f, g\rangle \in \mathrm{PR}(k+1)}$$

where

$$\mathrm{PR}(k) \quad := \quad \{f \in \mathrm{PR} \mid f : \mathbb{N}^k \to \mathbb{N}\}$$

Example 7.10. *Addition (+) and multiplication (×) on natural numbers are of course primitive recursive:*

$$\begin{array}{rcl} n+0 & = & n \\ n+(m+1) & = & (n+m)+1 \end{array} \qquad \begin{array}{rcl} n\times 0 & = & 0 \\ n\times(m+1) & = & n+(n\times m) \end{array}$$

Example 7.11. *The following primitive recursive functions come in particularly handy for intuitionistic arithmetic (see e.g. [vD04, §7.1] or [TvD88a, Chap. 3, §1.3]):*

$$\begin{array}{rclrcl} \overline{\mathrm{sg}}(0) & = & 1 & \overline{\mathrm{sg}}(n+1) & = & 0 \\ \mathrm{pred}(0) & = & 0 & \mathrm{pred}(n+1) & = & n \\ n \dot{-} 0 & = & n & n \dot{-} (m+1) & = & \mathrm{pred}(n \dot{-} m) \end{array}$$
$$|n-m| \quad = \quad (n \dot{-} m) + (m \dot{-} n)$$

Note that

$$n \dot{-} m \quad = \quad \begin{cases} n-m & \textit{if } n \geq m \\ 0 & \textit{otherwise} \end{cases}$$

so that $|n-m| = 0$ *iff* $n = m$.

Example 7.12. *Following [SU06, §A3] (also [TvD88a, Chap. 3, §1.8]) we assume a given primitive recursive bijection* $\langle -,- \rangle : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ *with primitive recursive inverses* π_1, π_2 *such that*

$$\pi_1\langle n,m\rangle \quad = \quad n \qquad \textit{and} \qquad \pi_2\langle n,m\rangle \quad = \quad m$$

Definition 7.13. *A relation* $\mathrm{P} \subseteq \mathbb{N}^k$ *is* ***primitive recursive*** *if its characteristic function*

$$\chi_{\mathrm{P}} \quad : \quad (n_1,\dots,n_k) \quad \longmapsto \quad \begin{cases} 1 & \textit{if } (n_1,\dots,n_k) \in \mathrm{P} \\ 0 & \textit{otherwise} \end{cases}$$

is primitive recursive.

Notation 7.14. *We often write* $\mathrm{P}(x_1,\dots,x_k)$ *to mean* $\mathrm{P} \subseteq \mathbb{N}^k$, *and* $\mathrm{P}(n_1,\dots,n_k)$ *(where* $n_1,\dots,n_k \in \mathbb{N}$*) to mean* $(n_1,\dots,n_k) \in \mathrm{P}$.

7.2.2. Representation of Computable Functions

Primitive recursion allows for a neat treatment of **Church-Turing computability**. Besides [SU06, §A3], we refer to *e.g.* [TvD88a, Chap. 2, §4 & §7], [vD04, §7.2] or [BBJ07, §6 & §8].

Theorem 7.15 (Kleene's T Predicate)**.** *There exist*

(a) for each partial recursive function $f : \mathbb{N} \to \mathbb{N}$, *a natural number* $e_f \in \mathbb{N}$ *(called the* ***Kleene index*** *of* f, *[vD04, §7.2]) and*

(b) a primitive recursive relation $\mathrm{T}(x,y,z,w)$

such that

- *for every partial recursive $f : \mathbb{N} \to \mathbb{N}$ and every $n, m \in \mathbb{N}$, we have $f(n) = m$ if and only if there is some $k \in \mathbb{N}$ such that $\mathrm{T}(e_f, n, m, k)$.*

Remark 7.16. *Theorem 7.15 can be obtained from the usual correspondence between partial recursive functions and Turing machines, together with the existence of a universal Turing machine (see e.g. [BBJ07, §6 & §8]). In particular, to each partial recursive function $f : \mathbb{N} \to \mathbb{N}$ corresponds a Turing machine M_f (see e.g. [BBJ07, Thm. 8.2]), and instead of the predicate $\mathrm{T}(x, y, z, w)$ of Thm. 7.15, one may have considered a primitive recursive predicate $T(x, y, z)$ such that*

- *$T(e_M, n, k)$ holds if and only if k codes a terminating computation of the Turing machine M on input n.*

Theorem 7.17 (The Halting Problem (Turing))**.** *The following problem is undecidable:*

- *Given a partial recursive function f and $n \in \mathbb{N}$, decide whether there exists some $m \in \mathbb{N}$ such that $f(n) = m$.*

Corollary 7.18. *The following problem is undecidable:*

- *Given a primitive recursive predicate $\mathrm{P}(x_1, \dots, x_k, y_1, \dots, y_\ell)$ and $n_1, \dots, n_k \in \mathbb{N}$, decide whether there exist some $m_1, \dots, m_\ell \in \mathbb{N}$ such that $\mathrm{P}(n_1, \dots, n_k, m_1, \dots, m_\ell)$.*

Proof. Exercise! □

7.2.3. The Language of First-Order Arithmetic

We consider a signature $\mathbf{\Sigma}_{\mathrm{PR}}$ for first-order arithmetic with one function symbol for each primitive recursive function. In §7.2.4 below, we shall then postulate appropriate (equational) axioms on these function symbols. But doing this in an effective way requires some care, essentially because testing equality of primitive recursive functions is undecidable (see Prop. 7.38, §7.2.7 below). We shall actually not merely assume one function symbol f for each primitive function $f \in \mathrm{PR}$, but rather assume one function symbol $\mathsf{f}_{\mathcal{T}}$ for each derivation tree $\mathcal{T}$ that ensures $f \in \mathrm{PR}$ according to the rules of Def. 7.9 (§7.2.1). The equational axioms on $\mathsf{f}_{\mathcal{T}}$ will then correspond to the equalities naturally associated to each rule of Def. 7.9 (see Notation 7.8).

We formally proceed as follows. Let T_{PR} be the set of finite derivation trees built according to the rules of Def. 7.9 (§7.2.1). Let $\mathcal{T} \in \mathrm{T}_{\mathrm{PR}}$ be such a derivation tree. By definition, $\mathcal{T}$ proves $f \in \mathrm{PR}(k)$ for some $f : \mathbb{N}^k \to \mathbb{N}$ and some $k \in \mathbb{N}$. We associate to $\mathcal{T}$ a **function symbol** $\mathsf{f}_{\mathcal{T}}$ of arity k.

We assume no predicate symbol besides equality (§6.4.1).

Definition 7.19. *The* ***signature*** $\mathbf{\Sigma}_{\mathrm{PR}}$ *is the one-sorted signature with no predicate symbol and with function symbols*

$$\begin{array}{lcl} \mathrm{Fun}(\mathbf{\Sigma}_{\mathrm{PR}}) & := & \bigcup_{k \in \mathbb{N}} \mathrm{Fun}(\mathbf{\Sigma}_{\mathrm{PR}})(k) \\ \mathrm{Fun}(\mathbf{\Sigma}_{\mathrm{PR}})(k) & := & \{\mathsf{f}_{\mathcal{T}} \mid \mathcal{T} \in \mathrm{T}_{\mathrm{PR}} \textit{ proves that } f \in \mathrm{PR}(k)\} \end{array}$$

The **language** of first-order arithmetic is the language of (one-sorted) first-order logic with equality over $\mathbf{\Sigma}_{\mathrm{PR}}$ (§6.4). In other words, assuming a set $\mathcal{V}$ of first-order variables, the formulae of first-order arithmetic are given by:

$$\begin{array}{rcl} A, B & ::= & (\mathsf{a} \doteq \mathsf{b}) \quad | \quad \top \quad | \quad \bot \quad | \quad A \wedge B \quad | \quad A \vee B \quad | \quad A \Rightarrow B \\ & | & (\forall \mathsf{x}) A \quad | \quad (\exists \mathsf{x}) A \end{array}$$

where $\mathsf{a}, \mathsf{b} \in \mathrm{Ter}(\mathbf{\Sigma}_{\mathrm{PR}}, \mathcal{V})$.

Notation 7.20.

(1) If $\mathcal{T}$ is

$$\frac{}{\mathrm{Zero}^0 \in \mathrm{PR}(0)}$$

then $\mathsf{f}_{\mathcal{T}}$ is written $\mathsf{0}$.

(2) If $\mathcal{T}$ is

$$\frac{}{\mathrm{Succ} \in \mathrm{PR}(1)}$$

then $\mathsf{f}_{\mathcal{T}}$ is written S.

(3) If $\mathcal{T}$ is

$$\frac{}{\pi^k_i \in \mathrm{PR}(k)}$$

then $\mathsf{f}_{\mathcal{T}}$ is written π^k_i.

Notation 7.21 (Numerals)**.** *Hence $\mathbf{\Sigma}_{\mathrm{PR}}$ is a signature with natural numbers in the sense of §7.1. In particular, we assume Notation 7.2, and to each $n \in \mathbb{N}$ corresponds the numeral $\underline{n} = \underbrace{\mathsf{S} \dots \mathsf{S}}_{n \text{ times}} \mathsf{0}$.*

7.2.4. The Axioms of First-Order Arithmetic

The axioms of first-order arithmetic $\mathbf{Ax}_{\mathrm{PR}}$ consist of the following rules.

(1) **Non-Confusion.**

$$\frac{}{\Delta \vdash (\forall \mathsf{x}) \neg (\mathsf{S}\mathsf{x} \doteq \mathsf{0})}$$

(2) **Induction Scheme.** For each formula A,

$$\frac{}{\Delta \vdash A[\mathsf{0}/\mathsf{x}] \;\Rightarrow\; (\forall \mathsf{y})(A[\mathsf{y}/\mathsf{x}] \Rightarrow A[\mathsf{S}\mathsf{y}/\mathsf{x}]) \;\Rightarrow\; (\forall \mathsf{x}) A}$$

(3) **Equational Axioms.** We associate equational axioms to trees $\mathcal{T} \in \mathrm{T}_{\mathrm{PR}}$ as follows.

- If $\mathcal{T}$ is

$$\frac{}{\mathrm{Zero}^k \in \mathrm{PR}(k)} \quad (k \geq 1)$$

then we have the axiom

$$\frac{}{\Delta \vdash (\forall \mathsf{x}_1, \dots, \mathsf{x}_k) \big(\mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k) \doteq \mathsf{0} \big)}$$

- If $\mathcal{T}$ is

$$\frac{}{\pi_i^k \in \mathrm{PR}(k)}$$

then we have the axiom

$$\frac{}{\Delta \vdash (\forall \mathsf{x}_1, \dots, \mathsf{x}_k)\big(\pi_i^k(\mathsf{x}_1, \dots, \mathsf{x}_k) \doteq \mathsf{x}_i\big)}$$

- If $\mathcal{T}$ is

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \mathcal{U}\end{array}}{g \in \mathrm{PR}(m)} \qquad \dfrac{\begin{array}{c}\vdots\\ \mathcal{V}_1\end{array}}{h_1 \in \mathrm{PR}(k)} \qquad \dots \qquad \dfrac{\begin{array}{c}\vdots\\ \mathcal{V}_m\end{array}}{h_m \in \mathrm{PR}(k)}}{g \circ \langle h_1, \dots, h_m \rangle \in \mathrm{PR}(k)}$$

then we have the axiom

$$\frac{}{\Delta \vdash (\forall \mathsf{x}_1, \dots, \mathsf{x}_k)\big(\mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k) \doteq \mathsf{f}_{\mathcal{U}}(\mathsf{f}_{\mathcal{V}_1}(\mathsf{x}_1, \dots, \mathsf{x}_k), \dots, \mathsf{f}_{\mathcal{V}_m}(\mathsf{x}_1, \dots, \mathsf{x}_k))\big)}$$

- If $\mathcal{T}$ is

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \mathcal{U}\end{array}}{g \in \mathrm{PR}(k)} \qquad \dfrac{\begin{array}{c}\vdots\\ \mathcal{V}\end{array}}{h \in \mathrm{PR}(k+2)}}{\mathrm{Iter}\langle g, h \rangle \in \mathrm{PR}(k+1)}$$

then we have the axioms

$$\frac{}{\Delta \vdash (\forall \mathsf{x}_1, \dots, \mathsf{x}_k)\big(\mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k, 0) \doteq \mathsf{f}_{\mathcal{U}}(\mathsf{x}_1, \dots, \mathsf{x}_k)\big)}$$

$$\frac{}{\Delta \vdash (\forall \mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y})\big(\mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{S}\mathsf{y}) \doteq \mathsf{f}_{\mathcal{V}}(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y}, \mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y}))\big)}$$

7.2.5. First-Order Peano Arithmetic

Definition 7.22 (First-Order Peano Arithmetic)**.** ***First-order Peano Arithmetic*** *(*PA*) is* NK$_1$ *with equality over* $(\mathbf{\Sigma}_{\mathrm{PR}}, \mathcal{V})$ *(§6.4.2) extended by all the rules of* $\mathbf{Ax}_{\mathrm{PR}}$*.*

Notation 7.23. *We also write* PA *for the* ***theory*** *induced by* PA*, i.e. the set of all closed formulae A such that $\vdash A$ in* PA*.*

Notation 7.24. *Once the axioms of first-order arithmetic are set, we notationaly forget about derivation trees $\mathcal{T}_f$ proving $f \in \mathrm{PR}$, and directly write f instead of $\mathcal{T}_f$. This in particular applies to all primitive recursive functions of §7.2.1 and §7.2.2.*

Example 7.25. *Recall the primitive recursive function* pred : $\mathbb{N} \to \mathbb{N}$ *of Ex. 7.11 (§7.2.1), with equations*

$$\mathrm{pred}(0) \quad = \quad 0 \qquad \textit{and} \qquad \mathrm{pred}(n+1) \quad = \quad n$$

The function pred *can be represented in* PA *via the tree*

$$\dfrac{\dfrac{}{\mathsf{0} \in \mathrm{PR}(0)} \qquad \dfrac{}{\pi^2_1 \in \mathrm{PR}(2)}}{\mathrm{Iter}\langle \mathsf{0}, \pi^2_1\rangle(x) \in \mathrm{PR}(1)}$$

so that the following rules are derivable in PA

$$\dfrac{}{\Delta \vdash \mathrm{pred}(\mathsf{0}) \doteq \mathsf{0}} \qquad \textit{and} \qquad \dfrac{}{\Delta \vdash (\forall \mathsf{x})\big(\mathrm{pred}(\mathsf{S}\mathsf{x}) \doteq \mathsf{x}\big)}$$

Remark 7.26. *Note that* $\mathbf{Ax}_{\mathrm{PR}}$ *does not include the rule*

$$\dfrac{}{\Delta \vdash (\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{S}\mathsf{x} \doteq \mathsf{S}\mathsf{y} \;\Rightarrow\; \mathsf{x} \doteq \mathsf{y}\big)}$$

expressing the injectivity of the successor function. But this rule is derivable from the equality axioms together with the rule

$$\dfrac{}{\Delta \vdash (\forall \mathsf{x})\big(\mathrm{pred}(\mathsf{S}\mathsf{x}) \doteq \mathsf{x}\big)}$$

of Ex. 7.25. In particular, PA *is a classical theory of natural numbers with induction in the sense of §7.1.*

Notation 7.27. *If* $\mathrm{P} \subseteq \mathbb{N}^k$ *is a primitive recursive relation (Def. 7.13, §7.2.1), we write* $\mathrm{P}(\mathsf{a}_1, \dots, \mathsf{a}_k)$ *for a formula* $\chi_{\mathrm{P}}(\mathsf{a}_1, \dots, \mathsf{a}_k) \doteq \underline{1}$ *(where* χ_{P} *is the characteristic function of* P*). We moreover call* $\mathrm{P}(x_1, \dots, x_k)$ *a primitive recursive predicate.*

Definition 7.28. *The* ***standard model*** *of* PA *is the set of natural numbers equipped with the obvious interpretation of each function symbol of* $\mathbf{\Sigma}_{\mathrm{PR}}$ *(so that each term* $\mathsf{a}(\mathsf{x}_1, \dots, \mathsf{x}_k)$ *induces a primitive recursive function* $[\![\mathsf{a}]\!] : \mathbb{N}^k \to \mathbb{N}$ *in the obvious way).*

A (closed) formula A *is* ***true*** *if* A *is valid (Def. 6.11 §6.1.3) in the standard model.*

Remark 7.29. *Our axiomatization of Peano Arithmetic is in accordance with [TvD88a, Chap. 3, §3.4] (see also [Koh08, §3.2]). But other usual definitions of* PA *do not include all primitive recursive functions from the start (besides [SU06, §9.2], see e.g. [vD04, §7.4]). Typically one considers (classical) first-order logic with equality over a (one-sorted) signature with natural numbers (§7.1) and binary function symbols* $(-) + (-)$, $(-) \times (-)$ *together with the axioms of* ***Non-Confusion*** *(7.2.4.(1)) and* ***Induction*** *(7.2.4.(2)) as well as the defining equality axioms for* $+, \times$ *(as in Ex. 7.10, §7.2.1). Write* PA_0 *for the resulting system.*

The difference is inessential since our version of PA *is* ***conservative*** *over* PA_0*, i.e. for every formula* A *of* PA_0*, we have* $\vdash A$ *in* PA_0 *if and only if* $\vdash A$ *in* PA *(Thm. 7.32, §7.2.6 below, see e.g. [SU06, Thm. 9.4.5]).*

7.2.6. Representation and Incompleteness

We now recall some well-known facts on the representation of functions in arithmetic. Of crucial importance is the notion of **provably total function** (Def. 7.31). But we shall also mention important facts of general interest, in particular Gödel's Incompleteness Theorems, and the conservativity of PA over PA_0 (Thm. 7.32), see Rem. 7.29 (§7.2.5) above. Most of this §7.2.6 is based on [SU06, §9], but we mention other pertinent references in the text.

We begin with the following usual fact on the representation of total recursive functions in (classical) arithmetic. Write $(\exists! \mathsf{z})B(\mathsf{z})$ for $(\exists \mathsf{z})\big(B(\mathsf{z}) \ \wedge \ (\forall \mathsf{w})(B(\mathsf{w}) \Rightarrow \mathsf{w} \doteq \mathsf{z})\big)$.

Theorem 7.30. *The following are equivalent for a function $f : \mathbb{N}^k \to \mathbb{N}$:*

- *The function f is total recursive.*
- *There exists a formula $A(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y})$ of PA_0 (with free variables as displayed) such that*
  - *(i) for all $n_1, \dots, n_k, m \in \mathbb{N}$, we have $f(n_1, \dots, n_k) = m$ if and only if PA_0 proves $\vdash A(\underline{n}_1, \dots, \underline{n}_k, \underline{m})$; and*
  - *(ii) PA_0 proves $\vdash (\forall \mathsf{x}_1 \cdots \mathsf{x}_n)(\exists! \mathsf{y})A(\mathsf{x}_1, \dots, \mathsf{x}_n, \mathsf{y})$.*

Theorem 7.30 is essentially due to Gödel, see [SU06, Prop. 9.4.2 & Thm. 9.4.3] (see also [vD04, §7.5]).

Beware that Thm. 7.30 does **not** mean that the classically provable $\forall\exists!$-statements of arithmetic can be witnessed by computable functions! The most fruitful notion of representation of function in arithmetic is the following. Recall Kleene's T predicate of Thm. 7.15 (7.2.2).

Definition 7.31 (Provably Total Function). *Let TH be a (classical or intuitionistic) theory over the signature $\mathbf{\Sigma}_{\mathrm{PR}}$ of arithmetic (Def. 7.19, §7.2.3). A partial recursive function $f : \mathbb{N} \to \mathbb{N}$ is **provably total** in TH if TH proves $(\forall \mathsf{x})(\exists \mathsf{z})\mathrm{T}(\underline{e}_f, \mathsf{x}, \pi_1 \mathsf{z}, \pi_2 \mathsf{z})$.*

While the notion of representation of Thm. 7.30 is not very attractive from the perspective of witness extraction (because of condition (i)), it (almost directly) gives the conservativity of PA_0 over PA (Rem. 7.29, §7.2.5), see [SU06, Thm. 9.4.5]

Theorem 7.32 (Conservativity)**.** *PA is conservative over PA_0.*

Remark 7.33. *Using [SU06, Thm. 9.4.4], Thm. 7.30 can be refined so as to attribute a PA_0-formula $A_{\mathsf{f}_{\mathcal{T}}}(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y})$ to each tree $\mathcal{T}$ proving $f \in \mathrm{PR}(k)$ in such a way that PA proves*

$$\vdash \quad (\forall \mathsf{x}_1 \dots \mathsf{x}_k)(\forall \mathsf{y})\big(\mathsf{f}_{\mathcal{T}}(\mathsf{x}_1, \dots, \mathsf{x}_k) \doteq \mathsf{y} \ \Leftrightarrow \ A_{\mathsf{f}_{\mathcal{T}}}(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y})\big)$$

*This in particular implies that PA proves all true equalities of the form $\mathsf{a} \doteq \mathsf{b}$ where a, b are **closed** terms.*

Theorem 7.30 is also crucial for Gödel's (first) Incompleteness Theorem 7.35, essentially because it implies that for every **recursive** $R \subseteq \mathbb{N}^k$, there is a formula $A(\mathsf{x}_1, \dots, \mathsf{x}_k)$ of PA_0 such that:

- for every $n_1, \dots, n_k \in \mathbb{N}$, we have $(n_1, \dots, n_k) \in R$ if and only if PA_0 proves $A(\underline{n}_1, \dots, \underline{n}_k)$.

Gödel's Incompleteness also relies on the arithmetization of syntax, *i.e.* the representation of syntactic objects (terms, formulae, proofs, etc.) by natural numbers. Besides [SU06, §9.3], see *e.g.* [vD04, §7.6] or [BBJ07, §15].

Theorem 7.34 (Gödel Numbers)**.** *There exist*

(a) a primitive recursive predicate $\mathrm{Proof}(x, y)$*, and*

*(b) for each formula A of PA_0 a natural number $\ulcorner A \urcorner$ (called the **Gödel number** of A, [SU06, §9.3])*

such that

- *for each formula A of PA_0, we have $\vdash A$ in PA_0 if and only if $\mathrm{Proof}(k, \ulcorner A \urcorner)$ for some $k \in \mathbb{N}$.*

The following is **Gödel's First Incompleteness Theorem** (see *e.g.* [vD04, §7.7] or [BBJ07, §17]).

Theorem 7.35 (Gödel)**.** *Assuming that PA_0 is consistent, there is a sentence Z such that PA_0 proves neither $\vdash Z$ nor $\vdash \neg Z$.*

Gödel's **Second Incompleteness Theorem** is a non-trivial strengthening of Thm. 7.35 according to which the undecidable sentence Z can be taken to be $(\forall \mathsf{z})\neg\mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$, where $\mathrm{Proof}(\mathsf{x}, \mathsf{y})$ stands for the formula of PA_0 which represents the primitive recursive relation Proof via Thm. 7.34. We refer to [BBJ07, Thm. 18.1] (see also [SU06, Thm. 9.3.3]).

Theorem 7.36 (Gödel)**.** *Assuming that PA_0 is consistent, neither $(\exists \mathsf{z})\mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$ nor $\neg(\exists \mathsf{z})\mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$ is provable in PA_0.*

It is immediate from the conservativity of PA over PA_0 (Thm. 7.32) that PA is also incomplete.

Corollary 7.37. *Assuming that PA is consistent, PA proves neither $(\exists \mathsf{z})\mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$ nor $\neg(\exists \mathsf{z})\mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$.*

7.2.7. A Classical Arithmetic of True Equalities

The definition of PA in §7.2.5 may seem a bit contrived because of the equational axioms 7.2.4.(3). It is tempting to rather consider the following system $\mathsf{PA}^\bullet$ of first-logic with equality over the signature with one function symbol f of arity k for each primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$. The axioms of $\mathsf{PA}^\bullet$ are **Non-Confusion** (7.2.4.(1)) and **Induction** (7.2.4.(2)), together with the following

$$\frac{[\![\mathsf{a}]\!] = [\![\mathsf{b}]\!]}{\Delta \vdash (\mathsf{x}_1, \dots, \mathsf{x}_k)\big(\mathsf{a} \doteq \mathsf{b}\big)}$$

where a, b have variables among $\mathsf{x}_1, \dots, \mathsf{x}_k$ and $[\![\mathsf{a}]\!], [\![\mathsf{b}]\!]$ stand for the obvious interpretations of a, b as primitive recursive functions $\mathbb{N}^k \to \mathbb{N}$.

Systems defined in this way may be interesting for concrete applications of proof-theory (see *e.g.* [Koh08, §3.5]). But beware that $\mathsf{PA}^\bullet$ is **not** conservative over PA_0 (assuming the consistency of PA_0), since $\mathsf{PA}^\bullet$ proves $(\forall \mathsf{z})\neg \mathrm{Proof}(\mathsf{z}, \ulcorner \bot \urcorner)$. Moreover, $\mathsf{PA}^\bullet$ has a non-recursive set of axioms because of the following.

Proposition 7.38. *The following problem is undecidable:*

- *Given primitive recursive functions $f, g : \mathbb{N}^k \to \mathbb{N}$, decide whether $f(n_1, \dots, n_k) = g(n_1, \dots, n_k)$ for all $n_1, \dots, n_k \in \mathbb{N}$.*

Proposition 7.38 essentially follows from the undecidability of the Halting Problem (Thm. 7.17, §7.2.2). But note that this requires an effective representation of the primitive recursive functions. Beware that Kleene's indexes (in the sense of Thm. 7.15, §7.2.2) are inappropriate for this since the set of Kleene's indexes of primitive recursive functions is **not** a recursive subset of the set of Kleene's indexes of all partial recursive functions (a direct consequence of **Rice's Theorem**, see *e.g.* [AC98, Cor. A1.3.2], see also [Odi99, Thm. II.2.9]). On the other hand, the construction of Gödel's numbers of formulae in Thm. 7.34 §7.2.6 (see *e.g.* [vD04, §7.6]) can readily be adapted to yield the following, based on the signature $\mathbf{\Sigma}_{\mathrm{PR}}$ of Def. 7.19 (§7.2.3).

Assumption 7.39. *We assume given*

(a) for each $\mathsf{f}_{\mathcal{T}} \in \mathrm{Fun}(\mathbf{\Sigma}_{\mathrm{PR}})(k)$, a natural number $\ulcorner \mathsf{f}_{\mathcal{T}} \urcorner$, and

(b) a primitive recursive function $\mathrm{const} : \mathbb{N} \to \mathbb{N}$, and

(c) for each $k, \ell \in \mathbb{N}$ a primitive recursive function $\mathrm{comp}_{k,\ell} : \mathbb{N}^{k+1} \to \mathbb{N}$

such that

(i) we have $\mathrm{const}(n) = \ulcorner \mathsf{f}_{\mathcal{T}} \urcorner$ where $\mathcal{T}$ proves $(- \mapsto \underline{n}) \in \mathrm{PR}(1)$, and

(ii) we have $\mathrm{comp}_{k,\ell}(\ulcorner \mathsf{f}_{\mathcal{T}} \urcorner, \ulcorner \mathsf{f}_{\mathcal{U}_1} \urcorner, \dots, \ulcorner \mathsf{f}_{\mathcal{U}_k} \urcorner) = \ulcorner \mathsf{f}_{\mathcal{V}} \urcorner$ where $\mathcal{V}$ proves $f \circ \langle g_1, \dots, g_k \rangle \in \mathrm{PR}(\ell)$ assuming $\mathcal{T}$ proves $f \in \mathrm{PR}(k)$ and $\mathcal{U}_i$ proves $g_i \in \mathrm{PR}(\ell)$ for $i = 1, \dots, k$.

We can now prove Prop. 7.38.

Proof. It clearly follows from Ex. 7.11 (§7.2.1) that we can consider the sub-problem of deciding whether $f(n_1, \dots, n_k) = 0$ for all $n_1, \dots, n_k \in \mathbb{N}$, in terms of the Gödel number $\ulcorner \mathsf{f}_{\mathcal{T}} \urcorner$ such that $\mathcal{T}$ proves $f \in \mathrm{PR}(k)$. Assume toward a contradiction that the problem is decidable for $k = 2$, so that we have a total recursive function $g : \mathbb{N} \to \mathbb{N}$ such that $g(\ulcorner \mathsf{f}_{\mathcal{T}} \urcorner) = 0$ iff $\mathcal{T}$ proves $f \in \mathrm{PR}(2)$ with $f(n, m) = 0$ for all $n, m \in \mathbb{N}$.

Let $\mathrm{t} : \mathbb{N}^4 \to \mathbb{N}$ be the characteristic function of Kleene's T predicate (Thm. 7.15, §7.2.2). For each $n, m \in \mathbb{N}$ the function $(k, \ell) \mapsto \mathrm{t}(n, m, k, \ell)$ is primitive recursive, and it follows from our Assumption 7.39 that the following function is primitive recursive:

$$h \quad : \quad (n, m) \quad \longmapsto \quad \ulcorner \mathsf{f}_{\mathcal{T}} \urcorner \text{ where } \mathcal{T} \text{ proves } \big((k, \ell) \mapsto \mathrm{t}(n, m, k, \ell)\big) \in \mathrm{PR}(2)$$

Hence the function

$$(n, m) \quad \longmapsto \quad (g \circ h)(n, m)$$

is total recursive.

Given a partial recursive $f : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$, we have

$$\begin{array}{lll} (g \circ h)(e_f, n) = 0 & \text{iff} & \mathrm{t}(e_f, n, -, -) \text{ is everywhere null} \\ & \text{iff} & \text{for all } m, k \in \mathbb{N} \text{ we have } (e_f, n, m, k) \notin \mathrm{T} \\ & \text{iff} & \text{for all } m \in \mathbb{N},\ f(n) \neq m \end{array}$$

But it is undecidable from e_f and n whether there is some $m \in \mathbb{N}$ such that $f(n) = m$. □

7.3. Heyting Arithmetic

We let Heyting Arithmetic (HA) be the intuitionistic version of Peano Arithmetic with function symbols for primitive recursive functions (PA, §7.2.5). This follows [SU06, §9.5] (see also [TvD88a, Chap. 3, §3], [Tro73, §1.3] and [Koh08, §3.2]).

Definition 7.40 (First-Order Heyting Arithmetic)**.** ***First-order Heyting Arithmetic*** *(*HA*) is* NJ$_1$ *with equality (§6.4.2) over the signature* $\mathbf{\Sigma}_{\mathrm{PR}}$ *(§7.2.3), extended by all the rules of* $\mathbf{Ax}_{\mathrm{PR}}$ *(§7.2.4).*

Notation 7.41. *Similarly as for Peano arithmetic (§7.2.5), we also write* HA *for the* ***theory*** *induced by* HA*, i.e. the set of all closed formulae* A *such that* $\vdash A$ *in* HA*.*

Sections 7.3.1 and 7.3.2 present basic properties of HA, essentially extending §7.1 with the benefits of having functions symbols for primitive recursive functions. The main properties of HA are then discussed in §7.3.3, most notably Kreisel's Theorem 7.54, according to which HA and PA prove the same "Π^0_2-sentences" (*i.e.* $\forall\exists$-sentences over primitive recursive predicates), and in particular prove the termination of the same algorithms. Finally, a proof of Kreisel's Theorem is presented in §7.3.4, using a technique originally due to Friedman.

7.3.1. Examples and Basic Properties

First note that Rem. 7.26 (§7.2.5) applies to HA, so that HA proves the axiom

$$(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{Sx} \doteq \mathsf{Sy} \ \Rightarrow\ \mathsf{x} \doteq \mathsf{y}\big)$$

of Fig. 19 (§7.1). Hence, HA is an intuitionistic theory of natural numbers with induction (Def. 7.1, §7.1). We thus get the following Prop. 7.42 and Prop. 7.43 from resp. Cor. 7.7 and Prop. 7.5 (§7.1).

Proposition 7.42. HA *proves the following:*

(1) $(\forall \mathsf{x})\big(\mathsf{x} \doteq 0 \ \vee\ \neg(\mathsf{x} \doteq 0)\big)$

(2) $(\forall \mathsf{x})\ \big(\mathsf{x} \doteq 0 \ \vee\ (\exists \mathsf{y})\big(\mathsf{x} \doteq \mathsf{Sy}\big)\big)$

(3) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x} \doteq \mathsf{y} \ \vee\ \neg(\mathsf{x} \doteq \mathsf{y})\big)$

Proposition 7.43. *Given formulae A, B, let*

$$A \mathbin{\dot\vee} B \quad := \quad (\exists \mathsf{z})\left[\big((\mathsf{z} \doteq 0) \Rightarrow A\big) \ \wedge\ \big(\neg(\mathsf{z} \doteq 0) \Rightarrow B\big)\right]$$

(where $\mathsf{z} \notin \mathrm{FV}(A) \cup \mathrm{FV}(B)$). Then HA *proves the following:*

(1) $A \Rightarrow A \mathbin{\dot\vee} B$

(2) $B \Rightarrow A \mathbin{\dot\vee} B$

(3) $(A \mathbin{\dot\vee} B) \ \Rightarrow\ (A \Rightarrow C) \ \Rightarrow\ (B \Rightarrow C) \ \Rightarrow\ C$

(4) $(A \mathbin{\dot\vee} B) \ \Leftrightarrow\ (A \vee B)$

Example 7.44. HA *has in particular function symbols for addition $(-)+(-)$ and multiplication $(-)\times(-)$ (Ex. 7.10, §7.2.1), on which* HA *proves*

$$\begin{array}{rcl} \mathsf{x}+0 & \doteq & \mathsf{x} \\ \mathsf{x}+\mathsf{S}(\mathsf{y}) & \doteq & \mathsf{S}(\mathsf{x}+\mathsf{y}) \end{array} \qquad\qquad \begin{array}{rcl} \mathsf{x}\times 0 & \doteq & 0 \\ \mathsf{x}\times\mathsf{S}(\mathsf{y}) & \doteq & \mathsf{x}+(\mathsf{x}\times\mathsf{y}) \end{array}$$

Moreover, HA *proves the following:*

(1) $(\forall \mathsf{y})(0+\mathsf{y} \doteq \mathsf{y})$

(2) $(\forall \mathsf{x})(\forall \mathsf{y})(\mathsf{x}+\mathsf{y} \doteq \mathsf{y}+\mathsf{x})$

(3) $(\forall \mathsf{x})(\forall \mathsf{y})(\forall \mathsf{z})\big((\mathsf{x}+\mathsf{y})+\mathsf{z} \doteq \mathsf{x}+(\mathsf{y}+\mathsf{z})\big)$

(4) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x}+\mathsf{y} \doteq 0 \ \Rightarrow\ \mathsf{y} \doteq 0\big)$

(5) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x}+\mathsf{y} \doteq \mathsf{y} \ \Rightarrow\ \mathsf{x} \doteq 0\big)$

Proof. Exercise! □

Definition 7.45. *Let*

$$\begin{array}{rcl} (\mathsf{x} \mathrel{\dot\leq} \mathsf{y}) & := & (\exists \mathsf{z})(\mathsf{x}+\mathsf{z} \doteq \mathsf{y}) \\ (\mathsf{x} \mathrel{\dot<} \mathsf{y}) & := & (\mathsf{x} \mathrel{\dot\leq} \mathsf{y}) \wedge \neg(\mathsf{x} \doteq \mathsf{y}) \end{array}$$

Lemma 7.46. HA *proves the following:*

(1) $(\forall \mathsf{x})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{x}\big)$

(2) $(\forall \mathsf{x})(\forall \mathsf{y})(\forall \mathsf{z})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{y} \ \Rightarrow\ \mathsf{y} \mathrel{\dot\leq} \mathsf{z} \ \Rightarrow\ \mathsf{x} \mathrel{\dot\leq} \mathsf{z}\big)$

(3) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{y} \ \Rightarrow\ \mathsf{y} \mathrel{\dot\leq} \mathsf{x} \ \Rightarrow\ \mathsf{x} \doteq \mathsf{y}\big)$

(4) $(\forall \mathsf{x})\neg\big(\mathsf{x} \mathrel{\dot<} \mathsf{x}\big)$

(5) $(\forall \mathsf{x})(\forall \mathsf{y})(\forall \mathsf{z})\big(\mathsf{x} \mathrel{\dot<} \mathsf{y} \ \Rightarrow\ \mathsf{y} \mathrel{\dot<} \mathsf{z} \ \Rightarrow\ \mathsf{x} \mathrel{\dot<} \mathsf{z}\big)$

(6) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{y} \;\Leftrightarrow\; (\mathsf{x} \doteq \mathsf{y} \;\vee\; \mathsf{x} \mathrel{\dot<} \mathsf{y})\big)$

(7) $(\forall \mathsf{x})\big(\mathsf{x} \mathrel{\dot\leq} 0 \;\Rightarrow\; \mathsf{x} \doteq 0\big)$

(8) $(\forall \mathsf{x})\neg\big(\mathsf{x} \mathrel{\dot<} 0\big)$

(9) $(\forall \mathsf{x})\big(0 \mathrel{\dot\leq} \mathsf{x}\big)$

(10) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{y} \;\Leftrightarrow\; \mathsf{Sx} \mathrel{\dot\leq} \mathsf{Sy}\big)$

(11) $(\forall \mathsf{x})\big(\mathsf{x} \mathrel{\dot<} \mathsf{Sx}\big)$

(12) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{y} \mathrel{\dot<} \mathsf{Sx} \;\Leftrightarrow\; \mathsf{y} \mathrel{\dot\leq} \mathsf{x}\big)$

(13) $(\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{x} \mathrel{\dot\leq} \mathsf{y} \;\vee\; \mathsf{x} \doteq y \;\vee\; \mathsf{y} \mathrel{\dot\leq} \mathsf{x}\big)$

Proof. Exercise! □

Proposition 7.47 (Well Founded Induction)**.** HA *proves the following:*

$$(\forall \mathsf{x})\,\big((\forall \mathsf{y})\big(\mathsf{y} \mathrel{\dot<} \mathsf{x} \;\Rightarrow\; A\mathsf{y}\big) \;\Rightarrow\; A\mathsf{x}\big) \;\Rightarrow\; (\forall \mathsf{x})A\mathsf{x}$$

Proof. Exercise! □

The following is [TvD88a, Chap. 3, §3.6] (see also the Mint-Orevkov Theorem [TvD88a, Chap. 2, §3.26]).

Proposition 7.48. HA *proves the following:*

(1) $(\exists \mathsf{x})A\mathsf{x} \;\Rightarrow\; \neg\neg(\exists \mathsf{x})\,\big(A\mathsf{x} \wedge (\forall \mathsf{z})\big(\mathsf{z} \mathrel{\dot<} \mathsf{x} \;\Rightarrow\; \neg A\mathsf{z}\big)\big)$

(2) $\neg\neg\,\big[(\exists \mathsf{x})A\mathsf{x} \;\Rightarrow\; (\exists \mathsf{x})\,\big(A\mathsf{x} \wedge (\forall \mathsf{z})\big(\mathsf{z} \mathrel{\dot<} \mathsf{x} \;\Rightarrow\; \neg A\mathsf{z}\big)\big)\big]$

Proof. Exercise! □

Corollary 7.49 (Minimum Principle)**.** PA *proves the following:*

$$(\exists \mathsf{x})A\mathsf{x} \;\Rightarrow\; (\exists \mathsf{x})\,\big(A\mathsf{x} \wedge (\forall \mathsf{z})\big(\mathsf{z} \mathrel{\dot<} \mathsf{x} \;\Rightarrow\; \neg A\mathsf{z}\big)\big)$$

7.3.2. The Quantifier-Free Formulae of HA

The following is essentially [Sim10, Def. I.7.8].

Definition 7.50 (The Arithmetical Hierarchy)**.**

(1) The ***quantifier-free*** *formulae of first-order arithmetic are given by the following grammar:*

$$A_0, B_0 \quad ::= \quad (\mathsf{a} \doteq \mathsf{b}) \quad | \quad \top \quad | \quad \bot \quad | \quad A_0 \wedge B_0 \quad | \quad A_0 \vee B_0 \quad | \quad A_0 \Rightarrow B_0$$

(2) A formula is Π^0_k (resp. Σ^0_k) if it is of the form

$$(\forall \vec{\mathsf{x}}_1)(\exists \vec{\mathsf{x}}_2)\cdots(\mathcal{Q}\vec{\mathsf{x}}_k)A_0 \qquad \textit{resp.} \quad (\exists \vec{\mathsf{x}}_1)(\forall \vec{\mathsf{x}}_2)\cdots(\mathcal{Q}\vec{\mathsf{x}}_k)A_0$$

where A_0 is quantifier-free and $\mathcal{Q}$ is either $\forall$ or $\exists$.

Remark 7.51.

(1) Following Notation 7.27 (§7.2.5), the quantifier-free formulae of arithmetic correspond exactly to the primitive recursive predicates (since the latter are closed under Boolean operations, see e.g. [vD04, Lem. 7.1.6]).

(2) Note that the parameter k in

$$(\forall \vec{\mathsf{x}}_1)(\exists \vec{\mathsf{x}}_2)\cdots(\mathcal{Q}\vec{\mathsf{x}}_k)A_0 \qquad \textit{and} \qquad (\exists \vec{\mathsf{x}}_1)(\forall \vec{\mathsf{x}}_2)\cdots(\mathcal{Q}\vec{\mathsf{x}}_k)A_0$$

*(where A_0 is quantifier-free) refers to the number of quantifier **alternations**, and is unrelated to the lengths of the vectors $\vec{\mathsf{x}}_i$.*

Actually, thanks to (primitive recursive) pairings (Ex. 7.12, §7.2.1), it would have been (provably in HA*) equivalent to define Π^0_k-formulae (resp. Σ^0_k-formulae) as the formulae of the form*

$$(\forall \mathsf{x}_1)(\exists \mathsf{x}_2)\cdots(\mathcal{Q}\mathsf{x}_k)A_0 \qquad \textit{resp.} \quad (\exists \mathsf{x}_1)(\forall \mathsf{x}_2)\cdots(\mathcal{Q}\mathsf{x}_k)A_0$$

(with A_0 quantifier-free).

*(3) The relevance of Def. 7.50 goes beyond first-order arithmetic (§**??**, see also e.g. [Sim10] as a whole). Of particular importance are the Σ^0_1-formulae and the Π^0_2-formulae, which are respectively of the form $(\exists \vec{\mathsf{x}})A_0$ and $(\forall \vec{\mathsf{x}})(\exists \vec{\mathsf{y}})B_0$ with A_0, B_0 quantifier-free, see §7.3.3 below.*

Lemma 7.52. *Recall the primitive recursive functions of Ex. 7.11 (§7.2.1). In* HA*, we have*

(1) $(\mathsf{a} \doteq \mathsf{b}) \;\Leftrightarrow\; |\mathsf{a} \dot{-} \mathsf{b}| \doteq 0$

(2) $(\mathsf{x} \doteq 0 \;\wedge\; \mathsf{y} \doteq 0) \;\Leftrightarrow\; \mathsf{x} + \mathsf{y} \doteq 0$

(3) $(\mathsf{x} \doteq 0 \;\vee\; \mathsf{y} \doteq 0) \;\Leftrightarrow\; \mathsf{x} \times \mathsf{y} \doteq 0$

(4) $(\mathsf{y} \doteq 0 \;\Rightarrow\; \mathsf{x} \doteq 0) \;\Leftrightarrow\; \overline{\mathrm{sg}}(\mathsf{y}) \times \mathsf{x} \doteq 0$

Corollary 7.53. *Let A_0 be a quantifier-free formula.*

(1) There is a first-order term a *such that* HA *proves*

$$\vdash \quad A_0 \quad \Leftrightarrow \quad (\mathsf{a} \doteq 0)$$

(2) HA *proves*

$$\vdash \quad A_0 \vee \neg A_0 \qquad \textit{and} \qquad \vdash \quad \neg\neg A_0 \Rightarrow A_0$$

7.3.3. Relation to PA and Extraction

Perhaps the most important property of HA is that it proves the same Π^0_2-formulae as PA. This result, originally due to Kreisel (see *e.g.* [TvD88a, Chap. 3, §10.6] or [SU06, 9.5.6]), shows in particular that HA proves the totality of the same partial recursive functions as PA (§7.2.2 and Def. 7.31, §7.2.6). In words, as alluded to in §1, when it comes to proving termination of algorithms, intuitionistic arithmetic is no weaker than classical arithmetic.

Theorem 7.54 (Kreisel)**.** *Let* $(\forall \mathsf{x})(\exists \mathsf{y})A_0$ *be a closed formula with* A_0 *quantifier-free. If* PA *proves* $(\forall \mathsf{x})(\exists \mathsf{y})A_0$ *then* HA *proves* $(\forall \mathsf{x})(\exists \mathsf{y})A_0$*.*

Corollary 7.55. PA *and* HA *proves the totality of the same partial recursive functions.*

Theorem 7.54 is *e.g.* [SU06, Thm. 9.5.6] (see also [TvD88a, Chap. 3, §3.4]). We shall prove it in §7.3.4 (Thm. 7.72).

Recall that since (primitive recursive) pairing and projections are available as first-order terms of PA and HA (Ex. 7.12, §7.2.1), Thm. 7.54 extends to closed formulae of the form $(\forall \mathsf{x}_1 \dots \mathsf{x}_n)(\exists \mathsf{y}_1 \dots \mathsf{y}_m)A_0$. One often says that Thm. 7.54 states the "Π^0_2-conservativity of PA over HA", *i.e.* that PA is conservative over HA for Π^0_2-sentences.

Theorem 7.54 extends to much stronger theories. We shall look at the case of Second-Order Arithmetic in §**??**. Harvey Friedman ([Fri78]) has shown that Cor. 7.55 also holds for **Set Theory**, namely w.r.t. an intuitionistic version IZF of ZF, see *e.g.* [Bee85, Chap. VIII XVI] (also [TvD88b, Chap. 11, §8.11, §9.6 & §9.7]).

Theorem 7.54 is often decomposed into the two following statements:

(1) For $A_0(\mathsf{x},\mathsf{y})$ quantifier-free, if $\mathsf{PA} \vdash (\forall \mathsf{x})(\exists \mathsf{y})A_0(\mathsf{x},\mathsf{y})$ then $\mathsf{HA} \vdash (\forall \mathsf{x})\neg\neg(\exists \mathsf{y})A_0(\mathsf{x},\mathsf{y})$.

(2) HA is closed under the rule

$$\frac{\vdash \neg\neg(\exists \mathsf{x})A_0}{\vdash (\exists \mathsf{x})A_0}\ (A_0 \text{ quantifier-free})$$

The rule in item (2) above is **Markov's Rule** (see *e.g.* [TvD88a, Chap. 3, §3.4] or [Koh08, §14]). Beware that it has an empty context! See Rem. 7.60 below for comments on the constructivity of Markov's Rule.

Markov's Rule and its close (but distinct) relative **Markov's Principle**

$$(\forall \mathsf{x})\big(A\mathsf{x} \vee \neg A\mathsf{x}\big) \quad \Rightarrow \quad \neg\neg(\exists \mathsf{x})A\mathsf{x} \quad \Rightarrow \quad A\mathsf{x}$$

(*e.g.* [Tro73, §1.11.5 §3.8]) have quite important roles in intuitionistic mathematics, see *e.g.* [TvD88a, Chap. 4, §5] or [Koh08] (as a whole).

Item (1) above can be obtained from the correctness of the negative translation $(-)^\neg$ (§6.4.4) of PA to HA.

Theorem 7.56. *If* PA *proves* A *then* HA *proves* $A^\neg$*.*

Theorem 7.56 easily gives Item (1) above. Indeed, from Thm 7.56 one gets that if PA proves $(\forall \mathsf{x})(\exists \mathsf{y})A_0$ with A_0 quantifier-free, then HA proves $(\forall \mathsf{x})\neg\neg(\exists \mathsf{y})A_0^{\neg}$, from which the decidability of quantifier-free formulae in HA (Cor. 7.53, §7.3.2) gives that HA proves $(\forall \mathsf{x})\neg\neg(\exists \mathsf{y})A_0$.

Note also that Thm. 7.56 reduces the consistency of PA to the consistency of HA.

Remark 7.57. *Theorem 7.56 implies that* HA *(as well as* PA*) prove all the true (closed)* Σ^0_1*-formulae. Beware that this does* **not** *hold for* Π^0_1*-formulae, because of Gödel's Second Incompleteness Theorem 7.36 (§7.2.6).*

Proof. Exercise! $\square$

Theorem 7.56 is proven exactly as for the case of first-order logic with equality (Thm. 6.65 §6.4.4 and Thm. 6.29 §6.2.3). See [SU06, Prop. 9.5.2] and [TvD88a, Chap. 3, §3.4] for variants (in the sense of Rem. 6.32, §6.2.3). We do not further comment on this, since our way to prove Kreisel's Theorem 7.54 in §7.3.4 will go via a generalization of the negative translation $(-)^{\neg}$ called **Friedman's Translation** (see Prop. 7.71).

Recall from §6.2.5 that Glivenko's Theorem 2.48 (§2.6) does not extend to the full language of first-order logic. In particular, we mentioned in Rem. 6.39 that not all formulae of the form

$$\neg\neg(\forall \mathsf{x}_1,\dots,\mathsf{x}_n)\big(A \vee \neg A\big)$$

are intuitionistically provable for the reason that there are consistent intuitionistic theories which prove some formulae of the form

$$\neg(\forall \mathsf{x}_1,\dots,\mathsf{x}_n)\big(A \vee \neg A\big)$$

We can now make this precise. Recall Kleene's T predicate from Thm. 7.15 (§7.2.2).

Proposition 7.58. *The extension of* HA *with the rule*

$$(\neg\forall\text{-EM}_{\mathrm{T}})\ \frac{}{\vdash \neg(\forall \mathsf{xy})\big((\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}') \vee \neg(\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}')\big)}$$

induces a consistent theory, i.e. the least intuitionistic theory containing HA *and the rule* $(\neg\forall\text{-EM}_{\mathrm{T}})$ *does not proves* $\bot$ *(see Def. 6.26, §6.2.2).*

In words, it is intuitionistically consistent to assume that not every computation either terminates or never halts! We shall prove Prop. 7.58 in §8.3.3 below (see also [TvD88a, Chap. 4, §3.4]). Note that Prop. 7.58 implies that there are instances of the **Double Negation Shift** which are not provable in HA (Rem. 6.38 and Prop. 6.37, §6.2.5).

The possibilities of extraction from proofs in HA may seem at first sight modest compared to the case of first-order logic (§6.2.6 and §6.4.6).

Theorem 7.59 (Extraction)**.** *In Heyting Arithmetic (*HA*),*

(1) from a proof of $\vdash A_1 \vee A_2$*,* ***with*** A_1, A_2 ***closed****, one can effectively compute an* $i \in \{1,2\}$ *and a proof of* $\vdash A_i$*;*

(2) there is a closed formula A such that $\vdash A \vee \neg A$ is not provable;

*(3) from a proof of $\vdash (\exists \mathsf{x})A$, **with** $(\exists \mathsf{x})A$ **closed**, one can effectively compute a natural number $n \in \mathbb{N}$ and a proof of $\vdash A[\underline{n}/\mathsf{x}]$.*

The conditions that the formulae are closed in item (1) and item (3) cannot be omitted. In contrast to the case of first-order logic (and recalling that disjunctions can be coded by existentials in HA (Prop. 7.43, §7.3.1)), witness extraction from proofs in arithmetic requires a proper extension of the term language. This is because the provably total functions (Def. 7.31, §7.2.6) of HA (equivalently of PA) are exactly those definable in a proper extension of the primitive recursive functions called **Gödel's System T** (§8).

Note that assuming item (1), Gödel's Incompleteness Theorems (in the form of Cor. 7.37, §7.2.6) directly give sentences A such that HA does not prove $A \vee \neg A$ (item (2)).

We shall prove Thm. 7.59 in §8.3.4 (Cor. 8.50). We refer to [TvD88a, Chap. 3, §5.6–5.10] (see also [SU06, Prop. 9.6.6]). See [TvD88a, Chap. 3, §10.5] for historical references.

Remark 7.60 (On Markov's Rule)**.** *Remark 7.57 immediately gives the closure of* HA *under the special case of Markov's Rule*

$$\frac{\vdash \neg\neg(\exists \mathsf{y})A_0(\mathsf{y})}{\vdash (\exists \mathsf{y})A_0(\mathsf{y})}$$

where $A_0(\mathsf{y})$ has at most y free.

The constructivity of Markov's Rule is often informally explained as follows (see e.g. [TvD88a, Chap. 4, §5.1]). Recall from Rem. 7.51 (§7.3.2) that a quantifier-free formula $A_0(\vec{\mathsf{x}})$ represents in the standard model (Def. 7.28, §7.2.5) a primitive recursive (and thus decidable) predicate. Consider a quantifier-free $A_0(\mathsf{x},\mathsf{y})$ with at most x,y free. From proof in HA *of*

$$\vdash \quad \neg\neg(\exists \mathsf{y})A_0(\mathsf{x},\mathsf{y})$$

we get for each $n \in \mathbb{N}$ a proof of the closed the formula $\neg\neg(\exists \mathsf{y})A_0(\underline{n},\mathsf{y})$, from which we know that there is some $m \in \mathbb{N}$ such that $A_0(\underline{n},\underline{m})$ is true (since we know it is impossible that there is none!). But (in view of Thm. 7.59.(3)), we might have no effective witness for the $(\exists \mathsf{y})$. However, since $A_0(x,y)$ is a decidable predicate, the function

$$\begin{array}{rcl} \mathbb{N} & \rightharpoonup & \mathbb{N} \\ n & \mapsto & \textit{the least } m \in \mathbb{N} \textit{ such that } A_0(\underline{n},\underline{m}) \textit{ is true} \end{array}$$

is total recursive. We can thus witness $(\exists \mathsf{y})A_0(\mathsf{x},\mathsf{y})$ by a computable function of x, but in general with no further property than the mere computability of this function.

7.3.4. Friedman's Translation

Our goal in this §7.3.4 is to prove Kreisel's Theorem 7.54 (§7.3.3). We shall use a technique originally due to Harvey Friedman, actually following the approach of [Miq11]. We first consider a notion of parametrized negation, which then induces a parametrized negative translation. See *e.g.* [SU06, Lem. 9.5.5], [TvD88a, Chap. 3, §5.1 – §5.5] or [Koh08, §14] for other presentations.

Definition 7.61 (Parametrized Negation)**.** *Let* $\mathtt{R}$ *be a formula of* HA*. The* $\mathtt{R}$*-**parametrized negation** is*

$$\neg_{\mathtt{R}} A \quad := \quad A \Rightarrow \mathtt{R}$$

We begin with some simple and basic properties of the parametrized negation.

Lemma 7.62. *The following are provable in* HA*.*

(1) $B \Rightarrow \neg_{\mathtt{R}} A \vdash A \Rightarrow \neg_{\mathtt{R}} B$

(2) $A \vdash \neg_{\mathtt{R}}\neg_{\mathtt{R}} A$

(3) $A \Rightarrow B \vdash \neg_{\mathtt{R}} B \Rightarrow \neg_{\mathtt{R}} A$

(4) $A \Rightarrow B \vdash \neg_{\mathtt{R}}\neg_{\mathtt{R}} A \Rightarrow \neg_{\mathtt{R}}\neg_{\mathtt{R}} B$

(5) $\neg_{\mathtt{R}}\neg_{\mathtt{R}}\neg_{\mathtt{R}} A \vdash \neg_{\mathtt{R}} A$

We now define the parametrized negative translation.

Definition 7.63 (Parametrized Negative Translation)**.** *Let* $\mathtt{R}$ *be a formula of* HA*. The **parametrized negative translation** of a formula* A *of* PA *is the formula* $A^{\neg_{\mathtt{R}}}$ *of* HA *defined by induction on* A *as follows:*

$$\begin{array}{rcl@{\qquad\qquad}rcl}
\bot^{\neg_{\mathtt{R}}} & := & \mathtt{R} & \top^{\neg_{\mathtt{R}}} & := & \top \\
(A \wedge B)^{\neg_{\mathtt{R}}} & := & A^{\neg_{\mathtt{R}}} \wedge B^{\neg_{\mathtt{R}}} & (\mathsf{a} \doteq \mathsf{b})^{\neg_{\mathtt{R}}} & := & \neg_{\mathtt{R}}\neg_{\mathtt{R}}(\mathsf{a} \doteq \mathsf{b}) \\
(A \vee B)^{\neg_{\mathtt{R}}} & := & \neg_{\mathtt{R}}(\neg_{\mathtt{R}} A^{\neg_{\mathtt{R}}} \wedge \neg_{\mathtt{R}} B^{\neg_{\mathtt{R}}}) & ((\forall \mathsf{x})A)^{\neg_{\mathtt{R}}} & := & (\forall \mathsf{x})A^{\neg_{\mathtt{R}}} \\
(A \Rightarrow B)^{\neg_{\mathtt{R}}} & := & A^{\neg_{\mathtt{R}}} \Rightarrow B^{\neg_{\mathtt{R}}} & ((\exists \mathsf{x})A)^{\neg_{\mathtt{R}}} & := & \neg_{\mathtt{R}}(\forall \mathsf{x})\neg_{\mathtt{R}} A^{\neg_{\mathtt{R}}}
\end{array}$$

Note that Def. 7.63 targets a fragment of HA without existential quantifiers nor disjunctions (except for those possibly occurring in $\mathtt{R}$). Hence, Def. 7.63 is (syntactically) independent from whether we take disjunctions as primitive or coded in HA (see §7.1 and Prop. 7.43, §7.3.1). For this reason, the remaining of this §7.3.4 is stated with a possible difference between the languages of HA and PA.

We are going to prove important basic properties of $(-)^{\neg_{\mathtt{R}}}$. We begin with simple facts. Note that $(\neg A)^{\neg_{\mathtt{R}}} = \neg_{\mathtt{R}} A^{\neg_{\mathtt{R}}}$.

Lemma 7.64. *The following are provable in* HA*, where* A*,* B *are formulae of* PA*:*

(1) $\vdash (A \Rightarrow A \vee B)^{\neg_{\mathtt{R}}}$

(2) $\vdash (A \vee \neg A)^{\neg_{\mathtt{R}}}$

(3) $\vdash (A[\mathsf{a}/\mathsf{x}] \Rightarrow (\exists \mathsf{x})A)^{\neg_{\mathtt{R}}}$

Proof. Exercise! □

We proceed with properties which require induction on formulae.

Lemma 7.65. *The following are provable in* HA*, where* A *is a formula of* PA*:*

(1) $\mathtt{R} \vdash A^{\neg_{\mathtt{R}}}$

(2) $\neg_{\mathtt{R}}\neg_{\mathtt{R}} A^{\neg_{\mathtt{R}}} \vdash A^{\neg_{\mathtt{R}}}$

Proof. Exercise! $\square$

For the soundness of $(-)^{\neg_{\mathtt{R}}}$ as a translation from classical to intuitionistic first-order logic, it is convenient to consider separately the following two properties.

Lemma 7.66. *The following are provable in* HA*, where* A*,* B *and* C *are formulae of* PA*, with* x *not free in* C *nor in* $\mathtt{R}$*:*

(1) $(A \vee B)^{\neg_{\mathtt{R}}},\ A^{\neg_{\mathtt{R}}} \Rightarrow C^{\neg_{\mathtt{R}}},\ B^{\neg_{\mathtt{R}}} \Rightarrow C^{\neg_{\mathtt{R}}} \vdash C^{\neg_{\mathtt{R}}}$

(2) $((\exists \mathsf{x})A)^{\neg_{\mathtt{R}}},\ (\forall \mathsf{x})(A^{\neg_{\mathtt{R}}} \Rightarrow C^{\neg_{\mathtt{R}}}) \vdash C^{\neg_{\mathtt{R}}}$

Proof. Exercise! $\square$

It is now direct to obtain the following first soundness property. If Δ is the context $A_1, \dots, A_m$, then we write $\Delta^{\neg_{\mathtt{R}}}$ for the context $A_1^{\neg_{\mathtt{R}}}, \dots, A_m^{\neg_{\mathtt{R}}}$.

Proposition 7.67. *If* $\Delta \vdash A$ *is provable in* NK_1 *then* $\Delta^{\neg_{\mathtt{R}}} \vdash A^{\neg_{\mathtt{R}}}$ *is provable in* NJ_1*.*

The proof of Prop. 7.67 is a direct extension of the corresponding property for the translation $(-)^{\neg}$ from NK_0 to NJ_0 (Thm. 2.39, §2.5). Modulo Rem. 6.32 (§6.2.3), the soundness of $(-)^{\neg}$ from NK_1 to NJ_1 (Thm. 6.29, §6.2.3) is essentially the instance of Prop. 7.67 with $\mathtt{R} := \bot$.

We now turn to the **Equality Rules** of Fig. 15 (§6.4.2), namely:

$$(\doteq\text{-I})\ \frac{}{\Delta \vdash \mathsf{a} \doteq \mathsf{a}} \qquad\qquad (\doteq\text{-E})\ \frac{\Delta \vdash \mathsf{a} \doteq \mathsf{b} \qquad \Delta \vdash A[\mathsf{a}/\mathsf{x}]}{\Delta \vdash A[\mathsf{b}/\mathsf{x}]}$$

Lemma 7.68. *The following rules are admissible in* NJ_1 *with equality (§6.4.2), where* $\mathsf{x} \notin \mathrm{FV}(\mathtt{R})$*:*

$$\frac{}{\Delta \vdash \neg_{\mathtt{R}}\neg_{\mathtt{R}}(\mathsf{a} \doteq \mathsf{a})} \qquad \textit{and} \qquad \frac{\Delta \vdash \neg_{\mathtt{R}}\neg_{\mathtt{R}}(\mathsf{a} \doteq \mathsf{b}) \qquad \Delta \vdash A^{\neg_{\mathtt{R}}}[\mathsf{a}/\mathsf{x}]}{\Delta \vdash A^{\neg_{\mathtt{R}}}[\mathsf{b}/\mathsf{x}]}$$

Proof. Exercise! $\square$

We thus get the extension of Prop. 7.67 to first-order logic with equality.

Corollary 7.69. *If* $\Delta \vdash A$ *is provable in* NK_1 *with equality, then* $\Delta^{\neg_{\mathtt{R}}} \vdash A^{\neg_{\mathtt{R}}}$ *is provable in* NJ_1 *with equality.*

The soundness of $(-)^{\neg}$ from NK_1 with equality to NJ_1 with equality (Thm. 6.65, §6.4.4) is essentially the instance of Cor. 7.69 with $\mathtt{R} := \bot$.

We finally turn to the axioms (*i.e.* non-logical rules) of PA (§7.2.4).

Lemma 7.70. *If* $\Delta \vdash A$ *is the conclusion of a rule of* $\mathbf{Ax}_{\mathrm{PR}}$ *(§7.2.4), where* Δ, A *are made of formulae of* PA*, then* $\Delta^{\neg_{\mathtt{R}}} \vdash A^{\neg_{\mathtt{R}}}$ *is provable in* HA*.*

Proof. Exercise! □

We now have everything we need to prove that $A^{\neg \mathtt{R}}$ is provable in HA whenever A is provable in PA.

Proposition 7.71. *If* $\Delta \vdash A$ *is provable in* PA *then* $\Delta^{\neg \mathtt{R}} \vdash A^{\neg \mathtt{R}}$ *is provable in* HA.

Modulo Rem. 6.32 (§6.2.3), the soundness of the negative translation $(-)^{\neg}$ from HA to PA (Thm. 7.56, §7.3.3) is Prop. 7.71 with $\mathtt{R} := \bot$.

We can finally obtain Kreisel's Theorem 7.54 (§7.3.3): PA is conservative over HA for Π^0_2-formulae.

Theorem 7.72 (Kreisel's Theorem 7.54)**.** *If* $(\forall \mathsf{x})(\exists \mathsf{y})(\mathsf{a} \doteq \mathsf{b})$ *is provable in* PA, *then it is provable in* HA.

The trick is to find a suitable (open) formula $\mathtt{R}$ for $((\forall \mathsf{x})(\exists \mathsf{y})(\mathsf{a} \doteq \mathsf{b}))^{\neg \mathtt{R}} \Rightarrow (\forall \mathsf{x})(\exists \mathsf{y})(\mathsf{a} \doteq \mathsf{b})$ to be provable in HA. Since formulae of the form $A^{\neg \mathtt{R}}$ contain no existential quantifiers (except possibly those of $\mathtt{R}$), it makes sense to think that $\mathtt{R}$ should contain one.

Proof. Exercise! □

8. Gödel's System T

When it comes to arithmetic, it is customary to give up looking for proof-terms in the sense of §6.3, and in particular for an exact representation of proofs as λ-terms as in Thm. 6.43. As discussed in §8.1 below, the difficulty comes from **Non Confusion** (7.2.4.(1)).

On the other hand, handling induction is not problematic, but leads to a proper extension of the simply-typed λ-calculus. The resulting calculus, known as **Gödel's System T**, actually characterizes exactly the provably total functions of HA (and thus of PA) (§8.2), and moreover leads to a BHK interpretation of HA thanks to a notion of **Realizability** (§8.3).

References to the literature are given within §8.2 and §8.3.

8.1. Proof-Terms for Induction

Let us look at possible proof-terms for intuitionistic arithmetic. For simplicity we consider the least intuitionistic theory of natural numbers with induction (§7.1), in which we moreover assume that disjunction is not primitive but coded using existentials as $A \dot{\vee} B$ (Prop. 7.5, §7.1).

So we might start from the following adaptation of §6.3 (and §6.4.5):

$$\begin{array}{rcl} t, u & ::= & x \mid \lambda x.t \mid tu \mid \langle t, u\rangle \mid \pi_1 t \mid \pi_2 t \\ & \mid & \langle\rangle \mid \mathtt{case}_\bot\ t\ \{\} \\ & \mid & \lambda \mathsf{x}.t \mid t\,\mathsf{a} \\ & \mid & \langle \mathsf{a}, t\rangle \mid \mathtt{let}\ \langle \mathsf{x}, y\rangle = t\ \mathtt{in}\ u \\ & \mid & \mathtt{eq}_I\ \mathsf{a} \mid \mathtt{eq}_E\ \mathsf{a}\ \mathsf{b}\ t\ u \end{array}$$

where a, b are given by the grammar

$$\mathsf{a}, \mathsf{b} \quad ::= \quad \mathsf{x} \quad | \quad \mathsf{0} \quad | \quad \mathsf{Sa}$$

It follows from §6.4.5 and §6.4.2 that via $\mathtt{eq}_I$ and $\mathtt{eq}_E$ we obtain suitable proof terms for the equality axioms of Fig. 20 (§7.1). So it remains to deal with the following:

$$\Delta \vdash (\forall \mathsf{x})\neg(\mathsf{Sx} \doteq \mathsf{0}) \qquad\qquad \Delta \vdash (\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{Sx} \doteq \mathsf{Sy} \;\Rightarrow\; \mathsf{x} \doteq \mathsf{y}\big)$$

$$\Delta \vdash A[\mathsf{0}/\mathsf{x}] \;\Rightarrow\; (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big) \;\Rightarrow\; (\forall \mathsf{x})A$$

We begin by the last one (*i.e.* the **Induction Scheme** 7.2.4.(2)), which we momentarily reformulate as the rule

$$\dfrac{\Delta \vdash A[\mathsf{0}/\mathsf{x}] \qquad \Delta \vdash (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}{\Delta \vdash (\forall \mathsf{x})A}$$

Following the methodology of §4.4 and §6.3.2, consider the following possible reductions in Natural Deduction.

Base Case:

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi_1}}{\Delta \vdash A[\mathsf{0}/\mathsf{x}]} \qquad \dfrac{\overset{\vdots}{\Pi_2}}{\Delta \vdash (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}}{\Delta \vdash (\forall \mathsf{x})A}}{\Delta \vdash A[\mathsf{0}/\mathsf{x}]} \qquad \rhd \qquad \dfrac{\overset{\vdots}{\Pi_1}}{\Delta \vdash A[\mathsf{0}/\mathsf{x}]}$$

Induction Step:

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi_1}}{\Delta \vdash A[\mathsf{0}/\mathsf{x}]} \qquad \dfrac{\overset{\vdots}{\Pi_2}}{\Delta \vdash (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}}{\Delta \vdash (\forall \mathsf{x})A}}{\Delta \vdash A[\mathsf{Sa}/\mathsf{x}]} \qquad \rhd$$

$$\dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi_2}}{\Delta \vdash (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}}{\Delta \vdash A[\mathsf{a}/\mathsf{x}] \Rightarrow A[\mathsf{Sa}/\mathsf{x}]} \qquad \dfrac{\dfrac{\dfrac{\overset{\vdots}{\Pi_1}}{\Delta \vdash A[\mathsf{0}/\mathsf{x}]} \qquad \dfrac{\overset{\vdots}{\Pi_2}}{\Delta \vdash (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}}{\Delta \vdash (\forall \mathsf{x})A}}{\Delta \vdash A[\mathsf{a}/\mathsf{x}]}}{\Delta \vdash A[\mathsf{Sa}/\mathsf{x}]}$$

This suggests to introduce a specific construction for induction at the level of proofs-terms. Recall the simple type A° associated to a formula A in §6.5 (on normalization of proof-terms for NJ_1). The simple types involved in the above induction rule are

$$\begin{array}{rcl}(A[\mathsf{0}/\mathsf{x}])^\circ & = & A^\circ \\ \big((\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)\big)^\circ & = & \mathtt{nat} \to (A^\circ \to A^\circ) \\ \big((\forall \mathsf{x})A\big)^\circ & = & \mathtt{nat} \to A^\circ\end{array}$$

where we write `nat` for the base type ι of §6.5. The proof-terms for induction shall be given by the **recursor** `Rec`, with (simple) typing

$$\frac{\mathcal{E} \vdash u : T \qquad \mathcal{E} \vdash v : \mathtt{nat} \to T \to T \qquad \mathcal{E} \vdash t : \mathtt{nat}}{\mathcal{E} \vdash \mathtt{Rec}(u, v, t) : T}$$

with reductions

$$\begin{array}{rcl}\mathtt{Rec}(u, v, \mathsf{0}) & \rhd_\beta & u \\ \mathtt{Rec}(u, v, \mathsf{S}t) & \rhd_\beta & v\, t\, \mathtt{Rec}(u, v, t)\end{array}$$

and with the following intended typing for proof-terms:

$$\frac{\mathcal{E} \vdash u : A[\mathsf{0}/\mathsf{x}] \qquad \mathcal{E} \vdash v : (\forall \mathsf{x})\big(A \Rightarrow A[\mathsf{Sx}/\mathsf{x}]\big)}{\mathcal{E} \vdash \lambda \mathsf{x}.\mathtt{Rec}(u, v, \mathsf{x}) : (\forall \mathsf{x})A}$$

We now come back to the two remaining axioms above, namely:

$$\Delta \vdash (\forall \mathsf{x})\neg(\mathsf{Sx} \doteq \mathsf{0}) \qquad \text{and} \qquad \Delta \vdash (\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{Sx} \doteq \mathsf{Sy} \;\Rightarrow\; \mathsf{x} \doteq \mathsf{y}\big)$$

Still following §6.5, the simple types involved are

$$\begin{array}{rcl}\big((\forall \mathsf{x})\neg(\mathsf{Sx} \doteq \mathsf{0})\big)^\circ & = & \mathtt{nat} \to \mathtt{unit} \to \mathtt{void} \\ \big((\forall \mathsf{x})(\forall \mathsf{y})\big(\mathsf{Sx} \doteq \mathsf{Sy} \;\Rightarrow\; \mathsf{x} \doteq \mathsf{y}\big)\big)^\circ & = & \mathtt{nat} \to \mathtt{nat} \to \mathtt{unit} \to \mathtt{unit}\end{array}$$

While it is not unreasonable to look for a proof-term for the injectivity of S, namely a closed λ-term of type

$$\mathtt{nat} \to \mathtt{nat} \to \mathtt{unit} \to \mathtt{unit}$$

there is no direct extension of proof-terms which would provide a closed λ-term of type

$$\mathtt{nat} \to \mathtt{unit} \to \mathtt{void}$$

without making the type `void` inhabited in the empty context, thus breaking consistency of the logical information carried-over by proof-terms (see Thm. 6.43, §6.3 and Thm. 6.55, §6.3.3).

$$(\mathtt{0})\ \frac{}{\mathcal{E} \vdash \mathtt{0} : \mathtt{nat}} \qquad\qquad (\mathtt{S})\ \frac{\mathcal{E} \vdash t : \mathtt{nat}}{\mathcal{E} \vdash \mathtt{S}\,t : \mathtt{nat}}$$

$$(\mathtt{Rec})\ \frac{\mathcal{E} \vdash u : T \qquad \mathcal{E} \vdash v : \mathtt{nat} \to T \to T \qquad \mathcal{E} \vdash t : \mathtt{nat}}{\mathcal{E} \vdash \mathtt{Rec}(u, v, t) : T}$$

Figure 21: Additional Typing Rules for System **T**.

8.2. Definition and Main Properties

This §8.2 focuses on the definition of Gödel's System **T** and on its main properties. Our main references are [SU06, §10] and [GLT89, §7]. Related material can be found in [TvD88b, Tro73, Koh08].

There are quite a lot of different variants of Gödel's System **T** considered in the literature (besides [SU06, GLT89], see *e.g.* [TvD88b, Tro73, Koh08]). We shall consider a version with (extended) λ-terms

$$\begin{array}{rcl} t, u & ::= & x \ \mid\ \lambda x.t \ \mid\ tu \ \mid\ \langle t, u\rangle \ \mid\ \pi_1 t \ \mid\ \pi_2 t \ \mid\ \langle\rangle \\ & \mid & \mathtt{0} \ \mid\ \mathtt{S}\,t \ \mid\ \mathtt{Rec}(u, v, t) \end{array}$$

As for typing, we consider types over the grammar

$$T, U \quad ::= \quad \mathtt{nat} \ \mid\ U \to T \ \mid\ T \times U \ \mid\ \mathtt{unit}$$

The typing rules consist of the rules of Fig. 6 (§4.4.3) for the above λ-terms and types, together with the additional typing rules of Fig. 21.

Notation 8.1.

(1) We often write $t : T$ for $\vdash t : T$.

*(2) Similarly as for the first-order terms of arithmetic (Notation 7.2, §7.1 and Notation 7.21, §7.2.3), to each $n \in \mathbb{N}$ corresponds as **numeral** $\underline{n} = \underbrace{\mathtt{S} \ldots \mathtt{S}}_{n \text{ times}} \mathtt{0}$.*

Remark 8.2. *Our choice of types (without* `void`*) implies that for each type T there is a closed term t_T of type T.*

Proof. Exercise! □

Concerning reduction, we consider the relation of β-reduction given by extending Fig. 5 (§4.4.3) with Fig. 22. More precisely, we first define the relation $\rhd_0$ with the basic rules of Fig. 22 (and Fig. 5), and then let $\rhd_\beta$ be the closure of $\rhd_0$ under the congruence rules of Fig. 22 (and Fig. 5).

The expressive power of System **T** is quite non-trivial. We come back on this in §8.2.3.

Basic Rules: extension of $\rhd_0$ (Fig. 5) with

$$\begin{array}{rcl} \mathtt{Rec}(u,v,\mathtt{0}) & \rhd_0 & u \\ \mathtt{Rec}(u,v,\mathtt{S}t) & \rhd_0 & v\,t\,\mathtt{Rec}(u,v,t) \end{array}$$

Congruence Rules: $\rhd_\beta$ is the least relation containing $\rhd_0$ and closed under the rules of Fig. 5 and

$$\frac{t \;\rhd_\beta\; t'}{\mathtt{S}\,t \;\rhd_\beta\; \mathtt{S}\,t'} \qquad \frac{u \;\rhd_\beta\; u'}{\mathtt{Rec}(u,v,t) \;\rhd_\beta\; \mathtt{Rec}(u',v,t)}$$

$$\frac{v \;\rhd_\beta\; v'}{\mathtt{Rec}(u,v,t) \;\rhd_\beta\; \mathtt{Rec}(u,v',t)} \qquad \frac{t \;\rhd_\beta\; t'}{\mathtt{Rec}(u,v,t) \;\rhd_\beta\; \mathtt{Rec}(u,v,t')}$$

Figure 22: Additional Reduction Rules for System **T**.

8.2.1. Structural Properties

System **T** enjoys the same basic structural properties as the typed λ-calculi seen earlier.

Lemma 8.3 (Structural Properties)**.**

(Weakening) *If* $\mathcal{E} \vdash t : T$ *and* $x \notin \mathrm{dom}(\mathcal{E})$ *then* $\mathcal{E}, x : U \vdash t : T$.

(Contraction) *If* $\mathcal{E}, x : U, y : U \vdash t : T$ *then* $\mathcal{E}, x : U \vdash t[x/y] : T$.

(Exchange) *If* $\mathcal{E}, x : U, \mathcal{E}', y : V, \mathcal{E}'' \vdash t : T$, *then* $\mathcal{E}, y : V, \mathcal{E}', x : U, \mathcal{E}'' \vdash t : T$.

Lemma 8.4 (Substitution)**.** *If* $\mathcal{E}, x : U \vdash t : T$ *and* $\mathcal{E} \vdash u : U$ *then* $\mathcal{E} \vdash t[u/x] : T$.

Proposition 8.5 (Subject Reduction)**.** *If* $\mathcal{E} \vdash t : T$ *and* $t \rhd_\beta u$ *then* $\mathcal{E} \vdash u : T$.

8.2.2. Normalization Properties

The normalization properties of System **T** are to all appearances similar to those of the typed λ-calculi seen earlier.

Theorem 8.6 (Strong Normalization)**.** *If* $\mathcal{E} \vdash t : T$ *then* t *is strongly* β*-normalizing.*

Theorem 8.7 (Confluence)**.** *For each typing context* $\mathcal{E}$ *and each type* T, *the relation* $\rhd_\beta$ *is confluent on* $|\mathcal{E} \vdash T|$ *(where* $|\mathcal{E} \vdash T|$ *is defined as in Rem. 4.11, §4.2.2).*

Lemma 8.8 (Closed Typed Normal Forms)**.** *If* t *is a closed typable term of System* **T** *in* β*-normal form, then* t *is of one of the following forms:*

$$\lambda x.u \qquad \langle\rangle \qquad \langle u, v\rangle \qquad \underline{n}$$

Proof. Exercise! □

Corollary 8.9. *If $\vdash t : \mathtt{nat}$, then there is a (unique) $n \in \mathbb{N}$ such that $t \rhd^*_\beta \underline{n}$.*

However, there is a crucial difference w.r.t. the λ-calculi seen earlier: the normalization of System **T** (actually already Cor. 8.9) **cannot** be proved in PA. We elaborate on this in §8.2.4 below. See [Tro73, §2.3.11] (see also [SU06, Cor. 10.4.11] for a related result). Just recall from §5.1 that the (strong) normalization of the simply-typed λ-calculus admits an arithmetic proof (see [SU06, §3.8] for references).

We now briefly discuss the adaptation of §5.3 to obtain Cor. 8.9. Strong Normalization (Thm. 8.6) is discussed separately in §8.2.5.

Corollary 8.9 follows from the adequacy of the following type interpretation. Given a type T, we write $|\vdash T|$ for the set of (closed) terms t such that $\vdash t : T$.

Definition 8.10 (Type Interpretation)**.** *For each type T we define a set $[\![T]\!] \subseteq |\vdash T|$ by induction on T as follows:*

$$\begin{array}{rcl} [\![\mathtt{nat}]\!] & := & \{t \in |\vdash \mathtt{nat}| \ ;\ t \rhd^*_\beta \underline{n} \text{ for some } n \in \mathbb{N}\} \\ [\![U \to T]\!] & := & \{t \in |\vdash U \to T| \ ;\ (\forall u \in [\![U]\!])(tu \in [\![T]\!])\} \\ [\![T \times U]\!] & := & \{t \in |\vdash T \times U| \ ;\ \pi_1 t \in [\![T]\!] \text{ and } \pi_2 t \in [\![U]\!]\} \end{array}$$

Lemma 8.11. *If $t \in [\![T]\!]$ and $u \rhd^*_\beta t$ with $u : T$, then $u \in [\![T]\!]$.*

Note that

- $\mathsf{0} \in [\![\mathtt{nat}]\!]$;
- $\mathsf{S}t \in [\![\mathtt{nat}]\!]$ for all $t \in [\![\mathtt{nat}]\!]$;
- and thus $\underline{n} \in [\![\mathtt{nat}]\!]$ for all $n \in \mathbb{N}$.

The following seemingly obvious property is essentially the cause of the unprovability of Cor. 8.9 in PA.

Lemma 8.12. *Assume $u \in [\![T]\!]$ and $v \in [\![\mathtt{nat} \to T \to T]\!]$. Then:*

(1) $\mathtt{Rec}(u, v, \underline{n}) \in [\![T]\!]$ *for all $n \in \mathbb{N}$.*

(2) $\mathtt{Rec}(u, v, t) \in [\![T]\!]$ *for all $t \in [\![\mathtt{nat}]\!]$.*

The adequacy of the type interpretation of Def. 8.10 is then an easy adaptation of §5.3.2 and §5.3.3.

Remark 8.13. *Similarly as in §5.3.1, Cor. 8.9 already holds for a weak variant of $\rhd_\beta$, namely the (deterministic) relation $\rhd_{\mathbf{T}}$ given by $E[t] \rhd_{\mathbf{T}} E[u]$ if $t \rhd_0 u$ (where $\rhd_0$ is given by Fig. 22) and $E[\ ]$ is a context on the grammar*

$$\begin{array}{rcl} E[\ ], F[\ ] & ::= & [\ ] \quad | \quad E[\ ]\, t \quad | \quad \pi_1\ E[\ ] \quad | \quad \pi_2\ E[\ ] \\ & | & \mathsf{S}\, E[\ ] \quad | \quad \mathtt{Rec}(u, v, F[\ ]) \end{array}$$

***where** $F[\ ]$ **is not of the form** $\mathsf{S}\, E[\ ]$.*

Proof. Exercise! □

8.2.3. Expressiveness and Representation

Maybe the most important property of Gödel's System **T** is that it exactly characterizes the provably total functions of HA (equivalently of PA).

We begin with some easy consequences of Cor. 8.9 (§8.2.2).

Notation 8.14 (Representation in System **T**)**.**

(1) Given $t : \mathtt{nat}$, *we write* $[t] \in \mathbb{N}$ *for the unique natural number such that* $t \rhd^*_\beta \underline{[t]}$.

(2) Given $t : \mathtt{nat}^k \to \mathtt{nat}$, *we write* $[t] : \mathbb{N}^k \to \mathbb{N}$ *for the function*

$$\begin{array}{rcl} \mathbb{N} \times \cdots \times \mathbb{N} & \longrightarrow & \mathbb{N} \\ (n_1, \dots, n_k) & \longmapsto & [t\langle \underline{n}_1 \dots \underline{n}_k \rangle] \end{array}$$

(3) We say that a function $f : \mathbb{N}^k \to \mathbb{N}$ *is* ***represented*** *by a term* $t : \mathtt{nat}^k \to \mathtt{nat}$ *if* $f = [t]$.

All primitive recursive functions (§7.2.1) are representable in System **T**.

Exercise 8.15. *Give a System* **T** *term* $\mathtt{add} : \mathtt{nat} \times \mathtt{nat} \to \mathtt{nat}$ *which represents addition on natural numbers.*

Proof. Exercise! $\square$

Proposition 8.16. *If* $f : \mathbb{N}^k \to \mathbb{N}$ *is primitive recursive, then there is a System* **T** *term* $t : \mathtt{nat}^k \to \mathtt{nat}$ *which represents* f.

Proof. Exercise! $\square$

However, System **T** represents more than the primitive recursive functions. We refer to [SU06, §10] and [GLT89, §7].

Exercise 8.17 (Ackermann Function)**.** *A typical function which is representable in System* **T** *but not primitive recursive is the well-known* ***Ackermann function****. We follow the presentation of [SU06, Ex. 10.3, 10.5 & 10.6], but see also [GLT89, §7.3.2].*

Consider the function $f : n \mapsto f_n(n)$, *where the functions* $(f_k : \mathbb{N} \to \mathbb{N})_{k \in \mathbb{N}}$ *are defined as*

$$\begin{array}{rcll} f_0(n) & = & n+1 & \\ f_{k+1}(n) & = & f_k^n(n) & \qquad (f_k \text{ applied } n \text{ times to } n) \end{array}$$

The function $f : n \mapsto f_n(n)$ *is not primitive recursive. Show that it is representable in System* **T**.

Proof. Exercise! $\square$

The main extraction result relating HA and System **T** is the following.

Theorem 8.18. *From a proof of* $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_k)(\exists \mathsf{y})A(\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y})$ *in* HA *(where* A *has free variables among* $\mathsf{x}_1, \dots, \mathsf{x}_k, \mathsf{y}$*), one can effectively compute a System* **T** *term* $t : \mathtt{nat}^k \to \mathtt{nat}$ *such that for all* $n_1, \dots, n_k \in \mathbb{N}$*, the formula* $A\big(\underline{n}_1, \dots, \underline{n}_k, \underline{[t\langle \underline{n}_1, \dots, \underline{n}_k\rangle]}\big)$ *is provable in* HA*.*

Corollary 8.19. *From a proof of* $(\forall \mathsf{x})(\exists \mathsf{y})\mathrm{P}(\mathsf{x}, \mathsf{y})$ *in* HA *(where* $\mathrm{P}(x, y)$ *is a primitive recursive predicate), one can effectively compute a System* **T** *term* t *such that* $\vdash t : \mathtt{nat} \to \mathtt{nat}$ *and* $\mathrm{P}(n, [t\underline{n}])$ *for all* $n \in \mathbb{N}$*.*

Corollary 8.19 of course extends to PA by Kreisel's Theorem 7.54 (§7.3.3). We shall prove Thm. 8.18 and Cor. 8.19 in §8.3 (Cor. 8.51 and Cor. 8.37 respectively) using a notation of **Realizability**.

Corollary 8.19 implies that the provably total functions of PA and HA (Def. 7.31, §7.2.6) are all representable in System **T**. The converse is actually true, as stated in the following.

Theorem 8.20 (Representation)**.** *The following are equivalent for a partial recursive function* $f : \mathbb{N} \to \mathbb{N}$*:*

(i) f *is provably total in* PA*;*

(ii) f *is provably total in* HA*;*

(iii) $f = [t]$ *for some term* t *such that* $\vdash t : \mathtt{nat} \to \mathtt{nat}$*.*

Theorem 8.20 is essentially [Tro73, Thm. 3.4.29]. It relies on tedious manipulations of Gödel's numbers representing System **T** terms in HA. We briefly comment on this in §8.2.4 below, but we refer to *e.g.* [Tro73, Thm. 3.4.29] for details (and to [GLT89, §7.4.2] for an informal account).

Let us finally mention that products types $((-) \times (-), \mathtt{unit})$ are not required for Cor. 8.19 and Thm. 8.20 (see Rem. 8.38, §8.3).

8.2.4. Arithmetization

This §8.2.4 overviews aspects of Gödel's System **T** related to its representation in first-order arithmetic via suitable Gödel's numbers. We use notations for these similar to those of §7.2.2 and §7.2.6 (see also §7.2.7). However, we shall not define the codings in full detail, and only loosely follow the approach of [vD04, §7.6] (on the representation of PA_0 in PA_0). We target two results:

(1) Proposition 8.23, namely the unprovablity in PA of the Normalization of System **T** (in the form of Cor. 8.9, §8.2.2), which is essentially a diagonalization argument based on extraction for Π^0_2-formulae (Cor. 8.19, §8.2.3) and using suitable Gödel numbers. (See [Tro73, §2.3.11], see also [SU06, Cor. 10.4.11] for a related result.)

(2) A sketch of the remaining part of Thm. 8.20 (§8.2.3), namely that the functions representable in System **T** are provable total in PA [Tro73, Thm. 3.4.29].

We consider a typed syntax for System **T**. First we assume for each type T a countably infinite set of variables $\mathcal{X}^T = \{x^T, y^T, z^T, \text{etc}\}$. The typed terms are then given by the following grammar, where types are indicated as superscripts:

$$\begin{array}{rcl} t^T, u^U & ::= & x^T \quad | \quad (\lambda x^U.t^T)^{U\to T} \quad | \quad (t^{U\to T}u^U)^T \\ & | & (\langle t^T, u^U\rangle)^{T\times U} \quad | \quad (\pi_1 t^{T\times U})^T \quad | \quad (\pi_2 t^{T\times U})^U \\ & | & \mathtt{0}^{\mathtt{nat}} \quad | \quad (\mathtt{S}t^{\mathtt{nat}})^{\mathtt{nat}} \quad | \quad (\mathtt{Rec}(u^T, v^{\mathtt{nat}\to T\to T}, t^{\mathtt{nat}}))^T \end{array}$$

Notation 8.21. *When possible, we drop the type superscript and write t for t^T. We also often write $t : T$ (or $\vdash t : T$) for t^T.*

Assumption 8.22. *We assume given*

(a) for each term t a Gödel number $\ulcorner t\urcorner \in \mathbb{N}$, for each type T a Gödel number $\ulcorner T\urcorner \in \mathbb{N}$, and

(b) primitive recursive functions $\mathrm{app}(x, y)$, $\mathrm{succ}(x)$, $\mathrm{num}(x)$ *and*

(c) primitive recursive predicates $\mathrm{Typing}(x, y, z)$, $\mathrm{Norm}(x, y, z)$

such that

(i) $\mathrm{app}(\ulcorner t\urcorner, \ulcorner u\urcorner) = \ulcorner tu\urcorner$, $\mathrm{succ}(\ulcorner t\urcorner) = \ulcorner \mathtt{S}t\urcorner$, $\mathrm{num}(n) = \ulcorner \underline{n}\urcorner$, *and*

(ii) $\mathrm{Typing}(\ulcorner t\urcorner, \ulcorner T\urcorner)$ *if and only if* $\vdash t : T$, *and*

(iii) $\mathrm{Norm}(\ulcorner t\urcorner, n, k)$ *if and only if* $t \rhd^k_\beta \underline{n}$.

Following *e.g.* [vD04, §7.6], we can assume that Gödel's numbers for types consist of pairs $\langle x_1, x_2\rangle$ where x_1 ranges among predefined integer values $\sharp\mathtt{c}$ for each constructor $\mathtt{c}$ of the syntax of types of System **T**, and where x_2 is a tuple whose length is the arity of the corresponding constructor and whose components are the codes of the corresponding sub-expressions. For terms, we assume a similar coding but this times as triples $\langle x_1, x_2, x_3\rangle$ where x_1 is ranges over predefined codes $\sharp\mathtt{c}$ for term constructors $\mathtt{c}$, x_2 is a tuple whose components are the codes of the sub-terms and where x_3 is the type. For instance:

$$\begin{array}{lcl} \ulcorner\mathtt{nat}\urcorner & = & \langle\sharp\mathtt{nat}, \langle\rangle\rangle \\ \ulcorner U\to T\urcorner & = & \langle\sharp{\to}, \langle\ulcorner U\urcorner, \ulcorner T\urcorner\rangle\rangle \\ \ulcorner\mathtt{0}\urcorner & = & \langle\sharp\mathtt{0}, \langle\rangle, \ulcorner\mathtt{nat}\urcorner\rangle \\ \ulcorner\mathtt{S}t\urcorner & = & \langle\sharp\mathtt{S}, \langle\ulcorner t\urcorner\rangle, \ulcorner\mathtt{nat}\urcorner\rangle \\ \ulcorner(tu)^T\urcorner & = & \langle\sharp @, \langle\ulcorner t\urcorner, \ulcorner u\urcorner\rangle, \ulcorner T\urcorner\rangle \end{array}$$

Moreover the primitives recursive functions app, succ and num are such that

$$\begin{array}{rcl} \mathrm{succ}(x) & = & \langle\sharp\mathtt{S}, \langle x\rangle, \ulcorner\mathtt{nat}\urcorner\rangle \\ \mathrm{num}(0) & = & \langle\sharp\mathtt{0}, \langle\rangle, \ulcorner\mathtt{nat}\urcorner\rangle \\ \mathrm{num}(\mathtt{S}(x)) & = & \langle\sharp\mathtt{S}, \langle\mathrm{num}(x)\rangle, \ulcorner\mathtt{nat}\urcorner\rangle \\ \mathrm{app}(x, y) & = & \langle\sharp @, \langle x, y\rangle, (\pi_2\circ\pi_2\circ\pi_3)(x)\rangle \end{array}$$

The predicate $\mathrm{Typing}(x, y)$ can be defined from the function $\mathrm{typeof}(x)$ with

$$\mathrm{typeof}(x) \quad = \quad \pi_3(x)$$

As for $\mathrm{Norm}(x, y, z)$, we can rely on Rem. 8.13 (§8.2.2) and instead of $\rhd_\beta$ consider the **deterministic** relation $\rhd_{\mathbf{T}}$ represented as a primitive recursive **function**.

Proposition 8.23. PA *does not prove* $(\forall \mathsf{x})(\exists \mathsf{z})\big(\mathrm{Typing}(\mathsf{x}, \ulcorner \mathtt{nat} \urcorner) \ \Rightarrow\ \mathrm{Norm}(\mathsf{x}, \pi_1 \mathsf{z}, \pi_2 \mathsf{z})\big)$.

Proof. Assume toward a contradiction that PA proves the formula. Consider the primitive recursive relation

$$\mathrm{P}(x, a, b, z) \quad := \quad \mathrm{Norm}\big(\mathrm{succ}(\mathrm{app}(x, \mathrm{num}(a))),\ b,\ z\big)$$

It follows from our Assumption 8.22 that

$$\begin{array}{rll} \mathrm{P}(\ulcorner u \urcorner, n, m, k) & \text{iff} & \mathrm{Norm}\big(\mathrm{succ}(\mathrm{app}(\ulcorner u \urcorner, \mathrm{num}(n))),\ m,\ k\big) \\ & \text{iff} & \mathrm{Norm}\big(\mathrm{succ}(\mathrm{app}(\ulcorner u \urcorner, \ulcorner \underline{n} \urcorner)),\ m,\ k\big) \\ & \text{iff} & \mathrm{Norm}\big(\mathrm{succ}(\ulcorner u\underline{n} \urcorner),\ m,\ k\big) \\ & \text{iff} & \mathrm{Norm}\big(\ulcorner \mathtt{S}(u\underline{n}) \urcorner,\ m,\ k\big) \\ & \text{iff} & \mathtt{S}(u\underline{n}) \rhd_\beta^k \underline{m} \end{array}$$

Our assumptions on Gödel numbers easily implies that PA proves the following

$$\begin{array}{rcl} & \vdash & \mathrm{Typing}\big(\mathrm{num}(\mathsf{x}),\ \ulcorner \mathtt{nat} \urcorner\big) \\ \mathrm{Typing}(\mathsf{x}, \ulcorner \mathtt{nat} \to \mathtt{nat} \urcorner) & \vdash & \mathrm{Typing}\big(\mathrm{app}(\mathsf{x}, \mathrm{num}(\mathsf{x})),\ \ulcorner \mathtt{nat} \urcorner\big) \\ \mathrm{Typing}(\mathsf{x}, \ulcorner \mathtt{nat} \to \mathtt{nat} \urcorner) & \vdash & \mathrm{Typing}\big(\mathrm{succ}(\mathrm{app}(\mathsf{x}, \mathrm{num}(\mathsf{x}))),\ \ulcorner \mathtt{nat} \urcorner\big) \end{array}$$

It follows that PA proves

$$(\forall \mathsf{x})(\exists \mathsf{z})\big(\mathrm{Typing}(\mathsf{x}, \ulcorner \mathtt{nat} \to \mathtt{nat} \urcorner) \ \Rightarrow\ \mathrm{P}(\mathsf{x}, \mathsf{x}, \pi_1 \mathsf{z}, \pi_2 \mathsf{z})\big)$$

Since the above relation is primitive recursive (as primitive recursive relations are closed under Boolean operations, see *e.g.* [vD04, Lem. 7.1.6]), it follows from Cor. 8.19 (§8.2.3) and from our Assumption 8.22 that there is a term $\vdash \mathtt{t} : \mathtt{nat} \to \mathtt{nat}$ such that for all $\vdash u : \mathtt{nat} \to \mathtt{nat}$ and all $n \in \mathbb{N}$, we have $\mathtt{S}(u\ \underline{\ulcorner u \urcorner}) \rhd_\beta^k \underline{m}$ for some $k \in \mathbb{N}$, where $m = [\mathtt{t}\,\underline{\ulcorner u \urcorner}]$. But this implies that $\mathtt{S}(\mathtt{t}\,\underline{\ulcorner \mathtt{t} \urcorner}) \rhd_\beta^k \underline{[\mathtt{t}\,\underline{\ulcorner \mathtt{t} \urcorner}]}$ for some $k \in \mathbb{N}$, a contradiction since $\mathtt{S}(\mathtt{t}\,\underline{\ulcorner \mathtt{t} \urcorner}) \rhd_\beta^* \mathtt{S}\underline{[\mathtt{t}\,\underline{\ulcorner \mathtt{t} \urcorner}]}$. $\square$

Extending Assumption 8.22 where necessary, we finally note here that we can define for each type T a formula $\mathsf{Norm}[\![T]\!]$ which represents in PA the set $[\![T]\!]$ (Def. 8.10, §8.2.2). Typical clauses are

$$\begin{array}{rcl} \mathsf{Norm}[\![\mathtt{nat}]\!](\mathsf{x}) & := & \mathrm{Typing}(\mathsf{x}, \ulcorner \mathtt{nat} \urcorner) \ \wedge\ (\exists \mathsf{y})(\exists \mathsf{z})\mathrm{Norm}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \\ \mathsf{Norm}[\![U \to T]\!](\mathsf{x}) & := & \mathrm{Typing}(\mathsf{x}, \ulcorner U \to T \urcorner) \\ & & \wedge\ (\forall \mathsf{y})\big(\mathsf{Norm}[\![U]\!](\mathsf{y}) \ \Rightarrow\ \mathsf{Norm}[\![T]\!](\mathrm{app}(\mathsf{x}, \mathsf{y}))\big) \end{array}$$

Note that (under Assumption 8.22) PA proves

$$\mathsf{Norm}[\![\mathtt{nat} \to \mathtt{nat}]\!](\mathsf{x}) \ \vdash\ (\forall \mathsf{y})(\exists \mathsf{z})(\exists \mathsf{w})\mathrm{Norm}\big(\mathrm{app}(\mathsf{x}, \mathrm{num}(\mathsf{y})),\ \mathsf{z},\ \mathsf{w}\big)$$

We can similarly write clauses expressing a form of higher-type computability. Possible clauses are

$$\begin{array}{rcl} \mathsf{Comput}[\![\mathtt{nat}]\!](\mathsf{x}) & := & \top \\ \mathsf{Comput}[\![U \to T]\!](\mathsf{x}) & := & (\forall \mathsf{y})\big(\mathsf{Comput}[\![U]\!](\mathsf{y}) \;\Rightarrow\; (\exists \mathsf{z})(\exists \mathsf{w})\big(\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{w}) \;\wedge\; \mathsf{Comput}[\![T]\!](\mathsf{z})\big)\big) \end{array}$$

where T is Kleene's predicate (Thm. 7.15, 7.2.2). Note that $\mathsf{Comput}[\![\mathtt{nat} \to \mathtt{nat}]\!](\mathsf{x})$ amounts to

$$(\forall \mathsf{y})(\exists \mathsf{z})(\exists \mathsf{w})\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{w})$$

The missing part of the Representation Theorem. 8.20 (§8.2.3) is obtained essentially by showing that for every $t : \mathtt{nat} \to \mathtt{nat}$, there is a partial recursive function $f : \mathbb{N} \to \mathbb{N}$ such that $[t] = f$ and such that PA proves $\mathsf{Comput}[\![\mathtt{nat} \to \mathtt{nat}]\!](\underline{e}_f)$. See [Tro73, Thm. 3.4.29] for details.

8.2.5. Strong Normalization

We briefly discuss here the adaptation of §5.5 to the strong normalization of (typed) System **T** terms (Thm. 8.6, §8.2.2).

We consider the following **elimination contexts**:

$$\begin{array}{rcl} E[\,], F[\,] \in \mathrm{Elim} & ::= & [\,] \quad | \quad E[\,]\,t \quad | \quad \pi_1\ E[\,] \quad | \quad \pi_2\ E[\,] \\ & | & \mathtt{Rec}(u, v, E[\,]) \end{array}$$

where (in contrast with Rem. 8.13, §8.2.2), we do not allow elimination contexts of the form $\mathtt{S}\,E[\,]$.

We consider the notion Weak Head Reduction $\rhd_{\mathrm{wh}}$ given by the direct application of Def. 5.4 (§5.3.1) to the above elimination contexts.

Recall the notion of tuple reduction of Notation 5.25 (§5.5.2). The analogue of the Weak Standardization Lemma 5.26 (§5.5.2) is the following:

Lemma 8.24 (Weak Standardization). *Let $E[\,] \in \mathrm{Elim}$.*

(1) If $E[(\lambda x.t)u] \rhd_\beta v$, then either $v = E[t[u/x]]$ or v is of the form $E'[(\lambda x.t')u']$ with $(E[\,], t, u) \rhd_\beta (E'[\,], t', u')$.

(2) Let $i \in \{1, 2\}$. If $E[\pi_i\langle t_1, t_2\rangle] \rhd_\beta v$, then either $v = E[t_i]$ or v is of the form $E'[\pi_i\langle t'_1, t'_2\rangle]$ with $(E[\,], t_1, t_2) \rhd_\beta (E'[\,], t'_1, t'_2)$.

(3) If $E[\mathtt{Rec}(u, v, \mathtt{0})] \rhd_\beta w$, then either $w = E[u]$ or w is of the form $E'[\mathtt{Rec}(u', v', \mathtt{0})]$ where $(E[\,], u, v) \rhd_\beta (E'[\,], u', v')$.

(4) If $E[\mathtt{Rec}(u, v, \mathtt{S}\,t)] \rhd_\beta w$, then either $w = E[v\,t\,\mathtt{Rec}(u, v, t)]$ or w is of the form $E'[\mathtt{Rec}(u', v', \mathtt{S}t')]$ where $(E[\,], u, v, t) \rhd_\beta (E'[\,], u', v', t')$.

Proof. Exercise! □

The Weak Head Expansion Lemma 5.27 is then adapted to the following.

Lemma 8.25 (Weak Head Expansion)**.** *Let $E[\ ] \in \mathrm{Elim}$.*

(1) If $E[t[u/x]] \in \mathcal{SN}$ and $u \in \mathcal{SN}$ then $E[(\lambda x.t)u] \in \mathcal{SN}$.

(2) Let $i \in \{1,2\}$. If $E[t_i] \in \mathcal{SN}$ and $t_{3-i} \in \mathcal{SN}$ then $E[\pi_i\langle t_1, t_2\rangle] \in \mathcal{SN}$.

(3) If $E[u] \in \mathcal{SN}$ and $v \in \mathcal{SN}$ then $E[\mathtt{Rec}(u, v, \mathtt{0})] \in \mathcal{SN}$.

(4) If $E[v\, t\, \mathtt{Rec}(u, v, t)] \in \mathcal{SN}$ then $E[\mathtt{Rec}(u, v, \mathtt{Rec}(u, v, \mathtt{S}\, t))] \in \mathcal{SN}$.

Proof. Exercise! □

This leads to the following notion of **Saturated Set** (Def. 5.28, §5.5.3).

Definition 8.26 (Saturated Sets)**.** *The set $\mathcal{SAT}$ of* ***saturated sets*** *is the set of all $A \subseteq \mathcal{SN}$ such that*

(i) $E[x] \in A$ whenever $E[\ ] \in \mathrm{Elim} \cap \mathcal{SN}$, and

(ii) $E[(\lambda x.t)u] \in A$ whenever $u \in \mathcal{SN}$ and $E[t[u/x]] \in A$, and

(iii) for all $i \in \{1,2\}$, $E[\pi_i\langle t_1, t_2\rangle] \in A$ whenever $t_{3-i} \in \mathcal{SN}$ and $E[t_i] \in A$, and

(iv) $E[\mathtt{Rec}(u, v, \mathtt{0})] \in A$ whenever $v \in \mathcal{SN}$ and $E[u] \in A$, and

(v) $E[\mathtt{Rec}(u, v, \mathtt{S}\, t)] \in A$ whenever $E[v\, t\, \mathtt{Rec}(u, v, t)] \in A$.

Lemma 8.27. *Saturated sets are stable under weak head expansion: given $A \in \mathcal{SAT}$ and $t \in A$, if $u \rhd_{\mathrm{wh}} t$ with $u \in \mathcal{SN}$ then $u \in A$.*

Proof. Exercise! □

Lemma 8.28. *The set $\mathcal{SAT}$ is a complete lattice w.r.t. inclusion, whose top element is $\top := \mathcal{SN}$ and whose least element is $\bot := \{t \in \mathcal{SN} \mid (\exists E[\ ], x)\big(t \rhd^*_{\mathrm{wh}} E[x]\big)\}$.*

We have the obvious analogue of Lem. 5.30 (§5.5.3).

Lemma 8.29. *Let $A, B \in \mathcal{SAT}$. Then*

$$A \mathrel{\Box\!\!\!\to} B,\ A \boxtimes B \ \in\ \mathcal{SAT}$$

Proof. Exercise! □

The interpretation of types shall follow §5 for $(-) \to (-)$ and $(-) \times (-)$. The type `nat` requires a specific fixpoint.

Definition 8.30. *We let $\mathcal{N}$ be the least fixpoint (in the complete lattice $\mathcal{SAT}$) of the (monotone) function*

$$\begin{array}{rcl} \mathcal{SAT} & \longrightarrow & \mathcal{SAT} \\ X & \longmapsto & \bot \cup \{t \in \mathcal{SN} \mid t \rhd^*_{\mathrm{wh}} \mathtt{0}\} \cup \{t \in \mathcal{SN} \mid (\exists u \in X)(t \rhd^*_{\mathrm{wh}} \mathtt{S}\, u)\} \end{array}$$

It is clear that $\mathtt{0} \in \mathcal{N}$ and that $\mathtt{S}\,t \in \mathcal{N}$ whenever $t \in \mathcal{N}$. Moreover,

Lemma 8.31. *Let $A \in \mathcal{SAT}$. Assume $u \in A$, $v \in (\mathcal{N} \boxminus\!\!\to A \boxminus\!\!\to A)$ and $t \in \mathcal{N}$. Then* $\mathtt{Rec}(u, v, t) \in A$.

PROOF. Exercise! □

We thus get an adequate type interpretations with

$$[\![\mathtt{nat}]\!] \;:=\; \mathcal{N} \qquad [\![U \to T]\!] \;:=\; [\![U]\!] \boxminus\!\!\to [\![T]\!] \qquad [\![T \times U]\!] \;:=\; [\![T]\!] \boxtimes [\![U]\!]$$

Strong normalization of the (typed) terms of System **T** (Thm. 8.6, §8.2.2) immediately follows.

8.3. Modified Realizability

Realizability is a technique introduced by S. C. Kleene as a formal counterpart to the informal Brouwer-Heyting-Kolmogorov (BHK) of intuitionistic arithmetic. Let us recapitulate the clauses of the BHK interpretation given in §4.1 and §6.3, but adapted to arithmetic (where the term "*proof*" should be understood as some informal notion of "witness of evidence"):

- there is no "*proof*" of $\bot$;
- a "*proof*" of $A_1 \wedge A_2$ is a pair of a "*proof*" of A_1 and a "*proof*" of A_2;
- a "*proof*" of $A_1 \vee A_2$ is a (dependent) pair of an $i \in \{1, 2\}$ and a "*proof*" of A_i;
- a "*proof*" of $A \Rightarrow B$ is a "function" taking a "*proof*" of A to a "*proof*" of B;
- a "*proof*" of $(\forall \mathsf{x})A$ is a "*function*" which takes a natural number $n \in \mathbb{N}$ to a "*proof*" of $A[\underline{n}/\mathsf{x}]$;
- a "*proof*" of $(\exists \mathsf{x})A$ is a (dependent) pair of a natural number $n \in \mathbb{N}$ and a "*proof*" of $A[\underline{n}/\mathsf{x}]$.

Kleene's original formulation of realizability was based on partial recursive functions, and interpreted intuitionistic arithmetic (which in particular allowed for the extraction of computable witnessing function from proofs of $\forall\exists$-formulae).

Nowadays, realizability exists in various forms. Besides Kleene's original version (for which we refer to [TvD88a, Chap. 4, §4], also [SU06, §9.6], and [Bee85, §VII]), the general methodology of realizability has been adapted to different computational models, *e.g.* computable functions on streams of natural numbers (see *e.g.* [Tro73, §3] or [KV65]), λ-calculi (or Combinatory Logics) either typed (*e.g.* [Tro73, §3.4] or [Koh08, §5]) or untyped (see *e.g.* [Kri93, §9]), as well as to different logics (we shall see the case of Second-Order Arithmetic in §**??** below). One of the most important development is Krivine's work on realizability for **classical** logic [Kri09], which actually goes as far as ZF set theory [Kri01].

$$\begin{array}{rcl}
(\mathsf{a} \doteq \mathsf{b})^\circ & := & \mathtt{unit} \\
\top^\circ & := & \mathtt{unit} \\
\bot^\circ & := & \mathtt{unit} \\
(A \Rightarrow B)^\circ & := & A^\circ \to B^\circ \\
(A \wedge B)^\circ & := & A^\circ \times B^\circ \\
((\forall \mathsf{x})A)^\circ & := & \mathtt{nat} \to A^\circ \\
((\exists \mathsf{x})A)^\circ & := & \mathtt{nat} \times A^\circ
\end{array}$$

Figure 23: Erasure Map from (Open) Formulae to Types.

$$\begin{array}{lll}
t \Vdash (\mathsf{a} \doteq \mathsf{b}) & \text{iff} & [\![\mathsf{a}]\!] = [\![\mathsf{b}]\!] \\
t \Vdash \top & & \\
t \nVdash \bot & & \\
t \Vdash A \Rightarrow B & \text{iff} & \text{for all } u \Vdash A, \text{ we have } tu \Vdash B \\
t \Vdash A \wedge B & \text{iff} & \pi_1 t \Vdash A \text{ and } \pi_2 t \Vdash B \\
t \Vdash (\forall \mathsf{x})A & \text{iff} & \text{for all } n \in \mathbb{N},\ t\underline{n} \Vdash A[\underline{n}/\mathsf{x}] \\
t \Vdash (\exists \mathsf{x})A & \text{iff} & \text{there exists } n \in \mathbb{N} \text{ such that } \pi_1 t \rhd_\beta^* \underline{n} \text{ and } \pi_2 t \Vdash A[\underline{n}/\mathsf{x}]
\end{array}$$

Figure 24: The Realizability Relation $t \Vdash A$ (where t, A are **closed** and $\vdash t : A^\circ$).

In any case, realizability is a very interesting computation-based tool to generate logical theories. We shall see in §8.3.3 below an example of a coherent intuitionistic theory which validates non-classical principles. But we also (again) refer to Krivine's work on models of ZF [Kri12].

In this §8.3, we concentrate on a simple variant of Kleene's Realizability, known as **Kreisel's Modified Realizability**, and which is based on System **T**. Besides [SU06, §10.4] (whose presentation largely differs from ours), we refer to [Koh08, §5] and [Tro73, §3.4] (which consider an internalized notion of (typed) realizability, based on Combinatory Logic).

The basic idea is a refinement of the above BHK interpretation, where a "*proof*" is taken to be a (closed) typed term of System **T**. We thus first associate to each (possibly open) formula A a type A°. We begin by only considering formulae of HA **without disjunction** (recall from §7.1 and Prop. 7.43, §7.3.1 that this is not a restriction in terms of expressiveness). The clauses for A° are given in Fig. 23. They should be strongly reminiscent from those of Fig. 17 (for the normalization of the proof-terms of NJ_1, §6.5). But **beware** that we now take

$$\bot^\circ \quad = \quad \mathtt{unit}$$

Definition 8.32 (Modified Realizability)**.** *The* ***realizability relation*** *$t \Vdash A$, where A is a closed formula and $t : A^\circ$, is defined by induction on A with the clauses of Fig. 24.*

We say that t ***realizes*** *A when $t \Vdash A$ and that A* ***is realized*** *when $t \Vdash A$ for some t.*

Example 8.33 (Non-Confusion)**.** *Taking* $\bot^\circ = \texttt{unit}$ *solves the problem raised in* §8.1 *with the* ***Non-Confusion*** *axiom* 7.2.4.(1)

$$(\forall \mathsf{x})\neg(\mathsf{S}\,\mathsf{x} \doteq \mathsf{0})$$

Indeed, we have

$$\lambda x.\lambda y.\langle\rangle \quad \Vdash \quad (\forall \mathsf{x})\big((\mathsf{S}\,\mathsf{x} \doteq \mathsf{0}) \;\Rightarrow\; \bot\big)$$

First, note that

$$\lambda x.\lambda y.\langle\rangle \quad : \quad \big((\forall \mathsf{x})\big((\mathsf{S}\,\mathsf{x} \doteq \mathsf{0}) \;\Rightarrow\; \bot\big)\big)^\circ \quad = \quad \texttt{nat} \to \texttt{unit} \to \texttt{unit}$$

Second, we have

$$\begin{array}{ll} & \lambda x.\lambda y.\langle\rangle \Vdash (\forall \mathsf{x})\big((\mathsf{S}\,\mathsf{x} \doteq \mathsf{0}) \;\Rightarrow\; \bot\big) \\ \textit{iff} & \textit{for all } n \in \mathbb{N},\ (\lambda x.\lambda y.\langle\rangle)\underline{n} \Vdash \big((\mathsf{S}\,\underline{n} \doteq \mathsf{0}) \;\Rightarrow\; \bot \\ \textit{iff} & \textit{for all } n \in \mathbb{N},\ \textit{for all } u \Vdash (\mathsf{S}\,\underline{n} \doteq \mathsf{0}),\ (\lambda x.\lambda y.\langle\rangle)\underline{n}u \Vdash \bot \end{array}$$

But given $n \in \mathbb{N}$, *we have* $(\lambda x.\lambda y.\langle\rangle)\underline{n}u \Vdash \bot$ *for all* $u : \texttt{unit}$ *such that* $u \Vdash \mathsf{S}\underline{n} \doteq \mathsf{0}$, *since there is no* $u : \texttt{unit}$ *such that* $u \Vdash \mathsf{S}\underline{n} \doteq \mathsf{0}$ *(as* $[\![\mathsf{S}\underline{n}]\!] = n + 1 \neq 0 = [\![\mathsf{0}]\!]$*).*

We now turn to a couple of customary properties of (modified) realizability.

Lemma 8.34. *Let* a, b *be* ***closed*** *first-order terms such that* $[\![\mathsf{a}]\!] = [\![\mathsf{b}]\!]$. *Then* $t \Vdash A[\mathsf{a}/\mathsf{x}]$ *if and only if* $t \Vdash A[\mathsf{b}/\mathsf{x}]$.

Lemma 8.35. *If* $t \Vdash A$ *and* $u \rhd_\beta t$ *with* $u : A^\circ$ *then* $u \Vdash A$.

The main result on (modified) realizability is that intuitionistically provable formulae are realized.

Theorem 8.36 (Adequacy)**.** *From a proof of* $\vdash A$ *in* HA *(with* A *closed), one can effectively compute a term* t *of System* **T** *such that* $t \Vdash A$.

The extraction of witnessing functions from proofs of Π^0_2-formulae (Cor. 8.19, §8.2.3) is a direct consequence of the Adequacy Theorem 8.36.

Corollary 8.37 (Cor. 8.19)**.** *From a proof of* $(\forall \mathsf{x})(\exists \mathsf{y})\mathrm{P}(\mathsf{x}, \mathsf{y})$ *in* HA *(where* $\mathrm{P}(x, y)$ *is a primitive recursive predicate), one can effectively compute a System* **T** *term* t *such that* $\vdash t : \texttt{nat} \to \texttt{nat}$ *and* $\mathrm{P}(n, [t\underline{n}])$ *for all* $n \in \mathbb{N}$.

Proof. By Def. 7.13 (§7.2.1), there is a first-order term $\mathsf{p}(x, y)$ such that HA proves $(\forall \mathsf{x})(\exists \mathsf{y})\mathsf{p}(\mathsf{x}, \mathsf{y}) \doteq \underline{1}$. By the Adequacy Theorem 8.36, there is a term $\vdash u : \texttt{nat} \to \texttt{nat} \times \texttt{unit}$ such that for all $n \in \mathbb{N}$, we have $\pi_2(u\underline{n}) \Vdash \mathsf{p}(\underline{n}, \underline{[\pi_1(u\underline{n})]}) \doteq \underline{1}$. Let $t := \lambda x.\pi_1(ux)$. By definition of realizability for equalities, we get that $\mathsf{p}(\underline{n}, \underline{[t\underline{n}]}) = 1$ for all $n \in \mathbb{N}$, and the result follows by Lem. 8.35. □

Remark 8.38. *Products types* $((-) \times (-), \texttt{unit})$ *are actually not required for Cor.* 8.37. *This can be seen via an appropriate notion of (modified) realizability, as presented in e.g.* [Koh08, §5].

The remaining of this §8.3 is organized as follows. First, we prove Thm. 8.36 in §8.3.1. An extension of modified realizability which handles (primitive) disjunctions is discussed in §8.3.2. Note that Thm. 8.36 means that (modified) realizability is compatible with intuitionistic deduction. In other words, the set of realized formulae is a (consistent) intuitionistic theory. We elaborate on this in §8.3.3. Finally, we discuss in §8.3.4 a variation on (modified) realizability which gives the extraction properties of HA (Thm. 7.59, §7.3.3 and Thm. 8.18, §8.2.3).

8.3.1. Proof of Adequacy

The inductive invariant of the Adequacy Theorem 8.36 is the following, where we use a vectorial notation for simultaneous substitution (§5.2), *e.g.* if $\vec{y} = y_1, \dots, y_n$ and $\vec{u} = u_1, \dots, u_n$, then $t[\vec{u}/\vec{y}]$ stands for $t[u_1/y_1, \dots, u_n/y_n]$.

Proposition 8.39. *Assume that $A_1, \dots, A_n, A$ have free (first-order) variables among $\mathsf{x}_1, \dots, \mathsf{x}_k$. From a proof of $A_1, \dots, A_n \vdash A$ in* HA*, one can effectively compute a term t of System* **T** *such that*

(i) $x_1 : \mathtt{nat}, \dots, x_k : \mathtt{nat}, y_1 : A_1^\circ, \dots, y_n : A_n^\circ \vdash t : A^\circ$, *and*

(ii) *for all* $m_1, \dots, m_k \in \mathbb{N}$ *and all* $u_1 \Vdash A_1[\underline{\vec{m}}/\vec{\mathsf{x}}], \dots, u_n \Vdash A_n[\underline{\vec{m}}/\vec{\mathsf{x}}]$, *we have* $t[\underline{\vec{m}}/\vec{x}, \vec{u}/\vec{y}] \Vdash A[\underline{\vec{m}}/\vec{\mathsf{x}}]$.

Notation 8.40. *We write $y_1 : A_1, \dots, y_n : A_n \vdash t \Vdash A$ when t satisfies conditions (i) and (ii) of Prop. 8.39.* ***Beware*** *that in contrast with Def. 8.32, this assumes neither t nor $A_1, \dots, A_n, A$ to be closed.*

Proposition 8.39 is proven by induction on derivations. We begin with the case of NJ_1 (without disjunction).

Lemma 8.41. *From a proof of $A_1, \dots, A_n \vdash A$ in NJ_1 (without disjunction), one can effectively compute a term t of System* **T** *such that $y_1 : A_1, \dots, y_n : A_n \vdash t \Vdash A$*

Proof. Exercise! □

We now turn to the **Equality Rules** of Fig. 15 (§6.4.2), namely:

$$(\doteq\text{-I})\ \frac{}{\Delta \vdash \mathsf{a} \doteq \mathsf{a}} \qquad\qquad (\doteq\text{-E})\ \frac{\Delta \vdash \mathsf{a} \doteq \mathsf{b} \qquad \Delta \vdash A[\mathsf{a}/\mathsf{x}]}{\Delta \vdash A[\mathsf{b}/\mathsf{x}]}$$

Via Ex. 6.60 (§6.4.2), we are done with the following.

Lemma 8.42. *Let a, b be* ***closed*** *first-order terms and let A be a formula with at most x free.*

(1) $\langle\rangle \Vdash \mathsf{a} \doteq \mathsf{a}$

(2) $\lambda e.\lambda x.x \Vdash \mathsf{a} \doteq \mathsf{b} \Rightarrow A[\mathsf{a}/\mathsf{x}] \Rightarrow A[\mathsf{b}/\mathsf{x}]$

Proof. Exercise! □

It remains to deal with the non-logical rules of HA (§7.2.4). The case of **Non-Confusion** 7.2.4.(1) was handled in Ex. 8.33, while each **Equational Axiom** 7.2.4.(3) is realized by a term of the form $\lambda\vec{x}.\langle\rangle$, where $\vec{x}$ has the same length as the corresponding prenex universal quantifier. We handle the **Induction Scheme** 7.2.4.(2) separately.

Lemma 8.43. *Let A be a formula with at most x free. Then*

$$\lambda y.\lambda z.\lambda x.\ \mathtt{Rec}(y,z,x) \quad\Vdash\quad A[\mathsf{0}/\mathsf{x}] \;\Rightarrow\; (\forall\mathsf{y})(A[\mathsf{y}/\mathsf{x}] \Rightarrow A[\mathsf{Sy}/\mathsf{x}]) \;\Rightarrow\; (\forall\mathsf{x})A$$

Proof. Exercise! □

This concludes the proof of Prop. 8.39 (and thus of Thm. 8.36 as well).

8.3.2. Extension to Primitive Disjunctions

It is actually possible to directly handle the full language of HA, without relying on the coded disjunction

$$A \dot{\vee} B \quad:=\quad (\exists\mathsf{z})\left[\left((\mathsf{z} \doteq \mathsf{0}) \Rightarrow A\right) \;\wedge\; \left(\neg(\mathsf{z} \doteq \mathsf{0}) \Rightarrow B\right)\right]$$

(neither at the level of formulae nor at the level of realizability). To this end, one can consider a version of System **T** with sum types. Its terms are given by

$$\begin{array}{rcl} t, u & ::= & x \mid \lambda x.t \mid tu \mid \langle t,u\rangle \mid \pi_1 t \mid \pi_2 t \mid \langle\rangle \\ & \mid & \mathtt{in}_1\, t \mid \mathtt{in}_2\, t \\ & \mid & \mathtt{case}\ t\ \{\mathtt{in}_1\, x_1 \mapsto u_1 \mid \mathtt{in}_2\, x_2 \mapsto u_2\} \\ & \mid & \mathtt{0} \mid \mathtt{S}\, t \mid \mathtt{Rec}(u,v,t) \end{array}$$

The types are

$$T, U \quad ::= \quad \mathtt{nat} \mid U \to T \mid T \times U \mid T + U \mid \mathtt{unit}$$

The erasure map $(-)^\circ$ is extended to

$$(A \vee B)^\circ \quad:=\quad A^\circ + B^\circ$$

and realizability for disjunctions is given by

$$t \Vdash A \vee B \quad \text{iff} \quad \text{there are } i \in \{1,2\} \text{ and } u_i \Vdash A_i \text{ with } t \rhd^*_\beta \mathtt{in}_i\, u_i \text{ and } u_i \Vdash A_i$$

The extension of Adequacy (§8.3.1) is given by the following.

Lemma 8.44. *Given **closed** formulae A_1, A_2, B, we have*

(1) $\lambda x.\, \mathtt{in}_i\, x \Vdash A_i \Rightarrow A_1 \vee A_2$

(2) $\lambda d.\lambda f_1.\lambda f_2.\ \mathtt{case}\ d\ \{\mathtt{in}_1\, x_1 \mapsto (f_1 x_1) \mid \mathtt{in}_2\, x_2 \mapsto (f_2 x_2)\}$
$\Vdash \quad A_1 \vee A_2 \;\Rightarrow\; (A_1 \Rightarrow B) \;\Rightarrow\; (A_2 \Rightarrow B) \;\Rightarrow\; B$

Proof. Exercise! □

Lemma 8.44 easily gives the extension of Prop. 8.39 (as well as Thm. 8.36) to formulae with primitive disjunctions. Note that the realizers in Lem. 8.44 depend only on A_i°, B°.

8.3.3. The Theory of Realized Formulae

An important consequence of the Adequacy Theorem 8.36 is that the set of closed formulae which are realized by (closed) terms of System **T** forms an intuitionistic theory in the sense of Def. 6.26 (§6.2.2). This theory is consistent since $\bot$ is not realized (by definition). But we shall see that (as claimed in Prop. 7.58, §7.3.3) it nevertheless validates principles which are classically false!

First note the following.

Lemma 8.45.

*(1) If A is a **closed** formula, then either A or $\neg A$ is realized, but not both.*

(2) If A is a closed Σ^0_1-formula (§7.3.2), then A is realizable if and only if A is true.

Proof. Exercise! $\square$

As a consequence of Lem. 8.45, we have that $A \vee \neg A$ is always realized for a closed formula A. One may thus think that the intuitionistic theories induced by realizability are in fact classical. But this is not the case, because Lem. 8.45 in general fails for open formulae, so that the universal closure of $A \vee \neg A$ is in general **not** realized, in the sense that assuming $\mathrm{FV}(A) \subseteq \{\mathsf{x}_1, \dots, \mathsf{x}_n\}$, we may have that

$$(\forall \mathsf{x}_1, \dots, \mathsf{x}_n)\big(A \ \vee \ \neg A\big)$$

has no realizer. In this case, by Lem. 8.45, the following formula is realized:

$$\neg(\forall \mathsf{x}_1, \dots, \mathsf{x}_n)\big(A \ \vee \ \neg A\big)$$

Proposition 8.46 (Prop. 7.58)**.** *The following formula is realized by a (closed) term of System* **T***:*

$$\neg(\forall \mathsf{xy})\big((\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}') \vee \neg(\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}')\big)$$

Proof. By definition of realizability for negated formulae, it is sufficient to show that the formula

$$(\forall \mathsf{xy})\big((\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}') \vee \neg(\exists \mathsf{zz}')\mathrm{T}(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{z}')\big)$$

is **not** realized. Assume toward a contradiction that it admits a realizer t in System **T**. Hence, given $n, m \in \mathbb{N}$,

- either $t\underline{n}\underline{m}$ reduces to $\mathtt{in}_1\, u_1$, and u_1 in turns give $k, \ell \in \mathbb{N}$ such that $\mathrm{T}(n,m,k,\ell)$;
- or $t\underline{n}\underline{m}$ reduces to $\mathtt{in}_2\, u_2$ where u_2 realizes $\neg(\exists \mathsf{zz}')\mathrm{T}(\underline{n},\underline{m},\mathsf{z},\mathsf{z}')$. But it follows from Lem. 8.45 that $(\exists \mathsf{zz}')\mathrm{T}(\underline{n},\underline{m},\mathsf{z},\mathsf{z}')$ is realized if and only if it is true, so that (by Lem. 8.45 again), the formula $\neg(\exists \mathsf{zz}')\mathrm{T}(\underline{n},\underline{m},\mathsf{z},\mathsf{z}')$ is realized if and only if it is true.

As a consequence, the function which takes e_f and m to the (weak head) normal form of $t\underline{e_f}\underline{m}$ decides the Halting Problem, a contradiction. $\square$

Corollary 8.47. *The following instance of the* ***Double Negation Shift*** *is* ***not*** *realized:*

$$(\forall \mathsf{x})\neg\neg\big((\exists \mathsf{z}\mathsf{z}')\mathrm{T}(\pi_1\mathsf{x},\pi_2\mathsf{x},\mathsf{z},\mathsf{z}') \vee \neg(\exists \mathsf{z}\mathsf{z}')\mathrm{T}(\pi_1\mathsf{x},\pi_2\mathsf{x},\mathsf{z},\mathsf{z}')\big) \quad \Rightarrow$$
$$\neg\neg(\forall \mathsf{x})\big((\exists \mathsf{z}\mathsf{z}')\mathrm{T}(\pi_1\mathsf{x},\pi_2\mathsf{x},\mathsf{z},\mathsf{z}') \vee \neg(\exists \mathsf{z}\mathsf{z}')\mathrm{T}(\pi_1\mathsf{x},\pi_2\mathsf{x},\mathsf{z},\mathsf{z}')\big)$$

Proof. Exercise! □

8.3.4. Realizability with Truth

It follows from Prop. 8.46 (§8.3.3) that not every realizable formula is provable. One can nevertheless obtain the Extraction Theorems for HA (Thm. 7.59, §7.3.3 and Thm. 8.18, §8.2.3) using a variant of (modified) realizability called (modified) **realizability with truth**, and denoted $t \Vdash_{\mathtt{t}} A$. The relation $t \Vdash_{\mathtt{t}} A$ is defined in the same way as $t \Vdash A$ (Fig. 24), but for the following clauses:

$$\begin{array}{lll} t \Vdash_{\mathtt{t}} A \Rightarrow B & \text{iff} & \mathsf{HA} \vdash A \Rightarrow B \text{ and for all } u \Vdash_{\mathtt{t}} A, \text{ we have } tu \Vdash_{\mathtt{t}} B \\ t \Vdash_{\mathtt{t}} (\forall \mathsf{x})A & \text{iff} & \mathsf{HA} \vdash (\forall \mathsf{x})A \text{ and for all } n \in \mathbb{N},\ t\underline{n} \Vdash_{\mathtt{t}} A[\underline{n}/\mathsf{x}] \end{array}$$

(recall that the involved formulae are all closed).

Lemma 8.48. *If $t \Vdash_{\mathtt{t}} A$ then* HA *proves $\vdash A$.*

Proof. Exercise! □

The extension of Adequacy (Thm. 8.36 and §8.3.1) is almost direct, but for the cases of of the rules ($\Rightarrow$-I) and ($\forall$-I) and of the non-logical rules of HA (§7.2.4).

Proposition 8.49 (Adequacy)**.** *From a proof of $\vdash A$ in* HA *(with A closed), one can effectively compute a term t of System* **T** *such that $t \Vdash_{\mathtt{t}} A$.*

Proof. Exercise! □

Corollary 8.50 (Extraction for HA (Thm. 7.59, §7.3.3))**.** *In Heyting Arithmetic (*HA*),*

(1) from a proof of $\vdash A_1 \vee A_2$, ***with*** *A_1, A_2* ***closed****, one can effectively compute an $i \in \{1,2\}$ and a proof of $\vdash A_i$;*

(2) from a proof of $\vdash (\exists \mathsf{x})A$, ***with*** *$(\exists \mathsf{x})A$* ***closed****, one can effectively compute a natural number $n \in \mathbb{N}$ and a proof of $\vdash A[\underline{n}/\mathsf{x}]$.*

Proof. Exercise! □

Corollary 8.51 (Extraction in System **T** (Thm. 8.18, §8.2.3))**.** *From a proof of $\vdash (\forall \mathsf{x}_1 \dots \mathsf{x}_k)(\exists \mathsf{y})A(\mathsf{x}_1,\dots,\mathsf{x}_k,\mathsf{y})$ in* HA *(where A has free variables among $\mathsf{x}_1,\dots,\mathsf{x}_k,\mathsf{y}$), one can effectively compute a System* **T** *term $t : \mathtt{nat}^k \to \mathtt{nat}$ such that for all $n_1,\dots,n_k \in \mathbb{N}$, the formula $A\big(\underline{n_1},\dots,\underline{n_k},\underline{[t\langle \underline{n_1},\dots,\underline{n_k}\rangle]}\big)$ is provable in* HA*.*

Proof. Exercise! □

An internalized version of (modified) realizability with truth is discussed in [Koh08, §5]. Corollary 8.50 can also be proven with a variant "with truth" of Kleene's Realizability (see *e.g.* [SU06, Prop. 9.6.6]). A somewhat different proof method is presented in [TvD88a, Chap. 3, §5.6–5.10].

9. Polymorphism

This §9 presents a typed λ-calculus introduced in the 70's independently by Girard (under the name **System F**) and by Reynolds (as the **Polymorphic λ-Calculus**). Girard's and Reynolds' motivations were completely different.

The motivation of Reynolds, coming from computer science, was to formalize the notion of **Parametric Polymorphism**, according to which extending simple types with type **variables** (ranged over by $X, Y, Z, \ldots \in \mathcal{V}_{\mathrm{Ty}}$) and **universal quantifications** on type variables (notation $(\forall X)T$) makes it possible to uniformly express that a single algorithm can be given several types. Typical examples (building on Ex. 3.1 §3.2.2 and to be developed in §9.1.3 and §9.2.2) are

$$\mathtt{id} \quad := \quad \lambda x.x \quad : \quad (\forall X)(X \to X)$$

as well as the usual polymorphic **map** function on lists

$$\mathtt{map} \quad : \quad (\forall X)(\forall Y)\big((X \to Y) \to \mathtt{list}(X) \to \mathtt{list}(Y)\big)$$

such that

$$\mathtt{map}\ f\ [t_0; \ldots ; t_n] \quad = \quad [f t_0; \ldots ; f t_n]$$

We refer to [Pie02, §23] for more on (parametric) polymorphism in (functional) programming languages.

Unlimited extension of simple types with type variables and (universal) type quantification results in a system with a huge expressive power. Besides (polymorphic) lists, one can represent any datatype over a finite signature (see *e.g.* [GLT89, §11.4]), as well as product and sum types (§4.4). From a logical perspective, the most important case is the polymorphic typing of **Church's numerals** (Ex. 3.1, §3.2.2), namely

$$\underline{n} \quad := \quad \lambda z.\lambda s.s^n z \quad : \quad \underbrace{(\forall X)\big(X \to (X \to X) \to X\big)}_{\mathtt{nat}}$$

(compare with the simple types of Ex. 4.3, §4.2.1).

Girard introduced System **F** with a purely logical motivation, namely as a (strongly) normalizing typed λ-calculus for **Second-Order Arithmetic** (§**??**). In particular, Girard has shown that the System **F** terms of type $\mathtt{nat} \to \mathtt{nat}$ (with $\mathtt{nat}$ as above) characterize exactly those partial recursive functions which are provably total in Second-Order Arithmetic (in very much the same way as Gödel's System **T** characterizes the provably total functions of First-Order Arithmetic (Thm. 8.20, §8.2.3)). See **??** below, as well as [SU06, §11 & §12] and [GLT89, §11 & §15].

As a consequence of Girard's characterization, the strong normalization of System **F** (a result also due to Girard, see [GLT89, §14]) **cannot** be proven in Second-Order Arithmetic (for very much the same reason as for System **T** w.r.t. First-Order arithmetic, §8.2.4). This results in a normalization proof which is bound to rely on principles going beyond second-order reasoning, and for which the **reducibility** machinery of §5 is the only proof method known yet.

System **F** usually comes under two different presentation, the so-called **Church-style** and **Curry-style** ones. In contrast with the simply-typed λ-calculi seen in §4 (but similarly as in the presentation of System **T** adopted in §8.2.4), in **Church-style** typed λ-calculi the whole typing derivations are recorded within the λ-terms. In the case of System **F**, this leads to a system with abstractions and applications for introduction and elimination of universal quantifications on types (in a manner reminiscent from the proof-terms for NJ_1 of §6.3). This setting has good type-checking properties, and is a convenient basis for further enrichment with **Dependent Types** (as in *e.g.* the **Calculus of Constructions**, see [SU06, §14] or *e.g.* [Bar92, §5]).

On the other hand, **Curry-style** systems record less typing information at the term level (compare the presentation of System **T** of §8.2 with that of §8.2.4). In the case of System **F**, this leads to a type system for the **pure** λ-terms of §3.2.1. Such settings are often better for studying the computational behavior of typed λ-terms, but typically lack decidable type-checking. Besides [SU06, §11.4] (on Curry-style System **F**), good further readings on Curry-style type systems are [Bar92, §4] and [Kri93].

We refer to [Bar92, §3] for a further discussion of Church-style vs Curry-style typing.

The remaining of this §9 is organized as follows. We present the (original) Church-style System **F** in §9.1, and the Curry-style version is discussed in §9.2. Finally, strong normalization (for both variants) is proven in §9.3.

Last but not least, logical aspects of System **F**, namely its strong ties with **Second-Order** logic, is discussed separately in §**??**.

9.1. Church-Style System F

Church-style System **F** (actually the original version of System **F**) is an explicit version of System **F**, in the sense that terms contain type annotations. This contrasts with the λ-calculus we have seen so far (which are called **Curry-style**). These annotations allow Church-style System **F** to have decidable type-checking (Rem. 9.22, §9.1.4).

Besides [SU06, §11.2], the standard reference is [GLT89, §11].

9.1.1. Definition

Fix a countably infinite set $\mathcal{V}_{\mathrm{Ty}} = \{X, Y, Z, \dots\}$ of **type variables**. The **types** of System **F** are given by the grammar

$$T, U \quad ::= \quad X \quad | \quad U \to T \quad | \quad (\forall X)T$$

where $X \in \mathcal{V}_{\mathrm{Ty}}$.

The **λ-terms** of **Church-Style** System **F** are given by the grammar

$$t, u \quad ::= \quad x \quad | \quad \lambda x : T.t \quad | \quad t\,u \quad | \quad \Lambda X.t \quad | \quad t\,T$$

where $X \in \mathcal{V}_{\mathrm{Ty}}$ and $x \in \mathcal{X}$ (§3.2.1).

Note that the syntax of **terms** depends on **types**, since besides **type abstractions** ($\Lambda X.t$) and **type applications** ($t\,T$), the type of the bound variable is recorded in the

Basic Rules:

$$(\lambda x : T.t)u \quad \rhd_0 \quad t[u/x] \qquad\qquad (\Lambda X.t)U \quad \rhd_0 \quad t[U/X]$$

Congruence Rules: $\rhd_\beta$ is the closure of $\rhd_0$ under the rules

$$\frac{t \;\rhd_\beta\; t'}{t\,u \;\rhd_\beta\; t'\,u} \qquad \frac{u \;\rhd_\beta\; u'}{t\,u \;\rhd_\beta\; t\,u'} \qquad \frac{t \;\rhd_\beta\; t'}{\lambda x : T.t \;\rhd_\beta\; \lambda x : T.t'}$$

$$\frac{t \;\rhd_\beta\; t'}{t\,U \;\rhd_\beta\; t'\,U} \qquad \frac{t \;\rhd_\beta\; t'}{\Lambda X.t \;\rhd_\beta\; \Lambda X.t'}$$

Figure 25: Beta-Reduction for Church-Style System **F**.

usual λ-abstraction

$$\lambda x : T.t$$

The notion of **free-variable** (Def. 3.3, §3.2.3) is adapted so as to encompass the free **type** variables occurring in a term and in a type.

Definition 9.1 (Free Variable)**.** *The sets* FV(T) *and* FV(t) *of* ***free*** *(term or type)* ***variables*** *of T and t are defined by induction as follows:*

$$\begin{array}{rcl} \mathrm{FV}(X) & := & \{X\} \\ \mathrm{FV}(U \to T) & := & \mathrm{FV}(U) \cup \mathrm{FV}(T) \\ & & \\ \mathrm{FV}((\forall X)T) & := & \mathrm{FV}(T) \setminus \{X\} \end{array} \qquad \begin{array}{rcl} \mathrm{FV}(x) & := & \{x\} \\ \mathrm{FV}(t\,u) & := & \mathrm{FV}(t) \cup \mathrm{FV}(u) \\ \mathrm{FV}(\lambda x : T.t) & := & \mathrm{FV}(T) \cup \mathrm{FV}(t) \setminus \{x\} \\ \mathrm{FV}(t\,T) & := & \mathrm{FV}(t) \cup \mathrm{FV}(T) \\ \mathrm{FV}(\Lambda X.t) & := & \mathrm{FV}(t) \setminus \{X\} \end{array}$$

Warning 9.2 (Variable Binding)**.** *As suggested by Def. 9.1 and similarly as for first-order formulae (§6.1.2), universal quantifiers in types $(\forall X)T$* ***bind*** *X in T. We thus assume that System* **F** *types are quotiented by α-equivalence (w.r.t. the obvious adaptation of §3.2.4). We also assume the obvious adaptation (from §6.1.2) of the notion of capture-avoiding type substitution.*

Concerning λ-terms, the variable X is ***bound*** *in $\Lambda X.t$, and we assume the corresponding extension of α-conversion and of capture-avoiding substitution (§3.2.3 and §3.3.1).*

The **full** (**strong**) relation $\rhd_\beta$ of β-reduction is defined in Fig. 25 as the closure of the **basic** reduction relation $\rhd_0$ under the indicated **congruence rules**.

The typing rules of (Church-style) System **F** are given in Fig 26, where given $\mathcal{E} = x_1 : T_1, \dots, x_n : T_n$, we write FV$(\mathcal{E})$ for FV$(T_1) \cup \cdots \cup$ FV(T_n).

9.1.2. Structural Properties

System **F** enjoys the same basic structural properties as the typed λ-calculi seen earlier.

$$(\textsc{Var})\ \frac{}{\mathcal{E} \vdash x : T}\ ((x : T) \in \mathcal{E})$$

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x : U.t : U \to T} \qquad (\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash t\,u : T}$$

$$(\forall^2\text{-I})\ \frac{\mathcal{E} \vdash t : T}{\mathcal{E} \vdash \Lambda X.t : (\forall X)T}\ (X \notin \mathrm{FV}(\mathcal{E})) \qquad (\forall^2\text{-E})\ \frac{\mathcal{E} \vdash t : (\forall X)T}{\mathcal{E} \vdash t\,U : T[U/X]}$$

Figure 26: Typing Rules of Church-Style System **F**.

Lemma 9.3 (Structural Properties)**.**

(Weakening) *If $\mathcal{E} \vdash t : T$ and $x \notin \mathrm{dom}(\mathcal{E})$ then $\mathcal{E}, x : U \vdash t : T$.*

(Contraction) *If $\mathcal{E}, x : U, y : U \vdash t : T$ then $\mathcal{E}, x : U \vdash t[x/y] : T$.*

(Exchange) *If $\mathcal{E}, x : U, \mathcal{E}', y : V, \mathcal{E}'' \vdash t : T$, then $\mathcal{E}, y : V, \mathcal{E}', x : U, \mathcal{E}'' \vdash t : T$.*

Lemma 9.4 (Substitution)**.**

(1) If $\mathcal{E}, x : U \vdash t : T$ and $\mathcal{E} \vdash u : U$ then $\mathcal{E} \vdash t[u/x] : T$.

(2) If $\mathcal{E} \vdash t : T$ then $\mathcal{E}[U/X] \vdash t[U/X] : T[U/X]$.

Proposition 9.5 (Subject Reduction)**.** *If $\mathcal{E} \vdash t : T$ and $t \rhd_\beta u$ then $\mathcal{E} \vdash u : T$.*

9.1.3. Examples of Impredicative Codings

We now turn to a series of examples of representation of various types in System **F**. All these examples strongly rely on (sometimes not prenex) universal type quantification, and for this reason are often called **impredicative** codings.

Most examples below (but for **lists**, 9.16–9.18) are impredicative representations of type constructions seen earlier in §4.4 and §8.2. We refer to [SU06, §11.3] for further examples (see also [GLT89, §11.3 – §11.5] and [Pie02, §23]).

Example 9.6 (Polymorphic Identity)**.** *Let*

$$\texttt{id} \quad := \quad \Lambda X.\ \lambda x : X.\ x \quad : \quad \underbrace{(\forall X)(X \to X)}_{\texttt{Id}}$$

For each type T we have

$$\texttt{id}\,T \quad \rhd_\beta \quad \lambda x : T.\ x \quad : \quad T \to T$$

and thus

$$\texttt{id}\,T\,t \quad \rhd_\beta \quad (\lambda x : T.\ x)\,t \quad \rhd_\beta \quad t$$

Note that

$$\texttt{id}\,\texttt{Id} \quad : \quad \texttt{Id} \to \texttt{Id}$$

Example 9.7 (Empty Type). *Let*

$$\texttt{void} \quad := \quad (\forall X)X$$

Note that

$$\frac{\mathcal{E} \vdash t : \texttt{void}}{\mathcal{E} \vdash t\,T : T}$$

We shall see in Prop. 9.24 (§9.1.4) that there is no closed term of type `void`*.*

Example 9.8 (Unit Type). *Let*

$$\texttt{unit} \quad := \quad (\forall X)(X \to X)$$

We have

$$\langle\rangle \quad := \quad \Lambda X.\ \lambda x : X.\ x \quad : \quad \texttt{unit}$$

Example 9.9 (Booleans). *Let*

$$\texttt{bool} \quad := \quad (\forall X)\big(X \to X \to X\big)$$

With

$$\begin{array}{rcl} \texttt{true} & := & \Lambda X.\ \lambda x : X.\ \lambda y : X.\ x \\ \texttt{false} & := & \Lambda X.\ \lambda x : X.\ \lambda y : X.\ y \\ (\texttt{if}\ t\ \texttt{then}\ u\ \texttt{else}\ v)_T & := & t\,T\,u\,v \end{array}$$

we have

$$\frac{}{\mathcal{E} \vdash \texttt{true} : \texttt{bool}} \qquad \frac{}{\mathcal{E} \vdash \texttt{false} : \texttt{bool}}$$

$$\frac{\mathcal{E} \vdash t : \texttt{bool} \qquad \mathcal{E} \vdash u : T \qquad \mathcal{E} \vdash v : T}{\mathcal{E} \vdash (\texttt{if}\ t\ \texttt{then}\ u\ \texttt{else}\ v)_T : T}$$

Moreover, we have

$$\begin{array}{rcccl} (\texttt{if true then}\ u\ \texttt{else}\ v)_T & = & \big(\Lambda X.\ \lambda x : X.\ \lambda y : X.\ x\big)\,T\,u\,v & \rhd^+_\beta & u \\ (\texttt{if false then}\ u\ \texttt{else}\ v)_T & = & \big(\Lambda X.\ \lambda x : X.\ \lambda y : X.\ y\big)\,T\,u\,v & \rhd^+_\beta & v \end{array}$$

Example 9.10 (Product Types). *Given types T, U, let*

$$T \times U \quad := \quad (\forall X)\big((T \to U \to X) \to X\big)$$

(where $X \notin \mathrm{FV}(T, U)$). Given λ-terms t, u, let

$$\begin{array}{rcl} \texttt{pair}\,t\,u & := & \Lambda X.\ \lambda p : T \to U \to X.\ p\,t\,u \\ \pi_1\,t & := & t\,T\,(\lambda x : T.\ \lambda y : U.\ x) \\ \pi_2\,t & := & t\,U\,(\lambda x : T.\ \lambda y : U.\ y) \end{array}$$

so that

$$\frac{\mathcal{E} \vdash t : T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash \texttt{pair}\,t\,u : T \times U} \qquad \frac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_1 t : T} \qquad \frac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_2 t : U}$$

Moreover, we have

$$\begin{array}{rcl}\pi_1(\mathtt{pair}\,t\,u) & = & \\ & \big(\Lambda X.\ \lambda p : T \to U \to X.\ p\,t\,u\big)\,T\,(\lambda x : T.\ \lambda y : U.\ x) & \rhd^+_\beta \\ & (\lambda x : T.\ \lambda y : U.\ x)\,t\,u & \rhd^+_\beta \quad t\end{array}$$

and similarly

$$\pi_2(\mathtt{pair}\,t\,u) \quad \rhd^+_\beta \quad u$$

Remark 9.11 (Polymorphic Products)**.** *Note that the terms* $\mathtt{pair}, \pi_1, \pi_2$ *of Ex. 9.10 actually **depend** on the types* T, U*. Let us make this precise by momentarily writing* $\mathtt{pair}^{T,U}, \pi_1^{T,U}, \pi_2^{T,U}$ *for the terms* $\mathtt{pair}, \pi_1, \pi_2$ *of Ex. 9.10.*

*One can actually define **polymorphic** versions of pairing and projections, namely (overriding the notations used in Ex. 9.10):*

$$\begin{array}{rcll}\mathtt{pair} & := & \Lambda Y.\Lambda Z.\lambda x : Y.\lambda y : Z.\ \underbrace{\Lambda X.\lambda p : Y \to Z \to X.\ p\,x\,y}_{\mathtt{pair}^{Y,Z} x\,y} & : \ (\forall YZ)\big(Y \to Z \to Y \times Z\big) \\ \pi_1 & := & \Lambda Y.\ \Lambda Z.\ \lambda p : Y \times Z.\ \underbrace{p\,Y\,(\lambda x : Y.\ \lambda y : Z.\ x)}_{\pi_1^{Y,Z} p} & : \ (\forall YZ)\big(Y \times Z \to Y\big) \\ \pi_2 & := & \Lambda Y.\ \Lambda Z.\ \lambda p : Y \times Z.\ \underbrace{p\,Z\,(\lambda x : Y.\ \lambda y : Z.\ y)}_{\pi_2^{Y,Z} p} & : \ (\forall YZ)\big(Y \times Z \to Z\big)\end{array}$$

Example 9.12 (Sum Types)**.** *Given types* T, U*, let*

$$T + U \quad := \quad (\forall X)\big((T \to X) \to (U \to X) \to X\big)$$

(where $X \notin \mathrm{FV}(T, U)$*). Given* λ*-terms* t, u*, let*

$$\begin{array}{rcl}\mathtt{inl}\,t & := & \Lambda X.\ \lambda \ell : T \to X.\ \lambda r : U \to X.\ \ell\,t \\ \mathtt{inr}\,u & := & \Lambda X.\ \lambda \ell : T \to X.\ \lambda r : U \to X.\ r\,u \\ \mathtt{case}\,V\,t\,u\,v & := & t\,V\,u\,v\end{array}$$

Note that

$$\frac{\mathcal{E} \vdash t : T}{\mathcal{E} \vdash \mathtt{inl}\,t : T + U} \qquad\qquad \frac{\mathcal{E} \vdash u : U}{\mathcal{E} \vdash \mathtt{inr}\,u : T + U}$$

$$\frac{\mathcal{E} \vdash t : T + U \qquad \mathcal{E} \vdash u : T \to V \qquad \mathcal{E} \vdash v : U \to V}{\mathcal{E} \vdash \mathtt{case}\,V\,t\,u\,v : V}$$

We have

$$\mathtt{case}\,V\,(\mathtt{inl}\,t)\,u\,v \quad \rhd^+_\beta \quad u\,t \qquad \textit{and} \qquad \mathtt{case}\,V\,(\mathtt{inr}\,t)\,u\,v \quad \rhd^+_\beta \quad v\,t$$

Remark 9.13 (Polymorphic Sums)**.** *Remark 9.11 extends to sum types, to the effect that we have polymorphic versions of* $\mathtt{inl}, \mathtt{inr}, \mathtt{case}$*:*

$$\begin{array}{rcl}\mathtt{inl} & : & (\forall XY)\big(Y \to X + Y\big) \\ \mathtt{inr} & : & (\forall XY)\big(Y \to X + Y\big) \\ \mathtt{case} & : & (\forall XYZ)\big((X + Y) \to (X \to Z) \to (Y \to Z) \to Z\big)\end{array}$$

Example 9.14 (Natural Numbers)**.** *Let*

$$\begin{array}{rcl}
\mathtt{nat} & := & (\forall X)\big(X \to (X \to X) \to X\big) \\
\underline{n} & := & \Lambda X.\ \lambda z : X.\ \lambda s : X \to X.\ s^n z \\
\mathtt{Z} & := & \Lambda X.\ \lambda z : X.\ \lambda s : X \to X.\ z \\
\mathtt{S}\, t & := & \Lambda X.\ \lambda z : X.\ \lambda s : X \to X.\ s\,(t\,X\,z\,s) \\
\mathtt{iter}\, U\, t\, u\, v & := & t\, U\, u\, v
\end{array}$$

(where $n \in \mathbb{N}$). Note that

$$\frac{}{\mathcal{E} \vdash \mathtt{Z} : \mathtt{nat}} \qquad \frac{\mathcal{E} \vdash t : \mathtt{nat}}{\mathcal{E} \vdash \mathtt{S}\, t : \mathtt{nat}} \qquad \frac{}{\mathcal{E} \vdash \underline{n} : \mathtt{nat}}\ (n \in \mathbb{N})$$

$$\frac{\mathcal{E} \vdash t : \mathtt{nat} \qquad \mathcal{E} \vdash u : U \qquad \mathcal{E} \vdash v : U \to U}{\mathcal{E} \vdash \mathtt{iter}\, U\, t\, u\, v : U}$$

We have

$$\begin{array}{rcl}
\mathtt{S}\, \underline{n} & \rhd_\beta^+ & \underline{n+1} \\
\mathtt{iter}\, U\, \mathtt{Z}\, u\, v & \rhd_\beta^+ & u \\
\mathtt{iter}\, U\, (\mathtt{S}\, t)\, u\, v & =_\beta & v\,(\mathtt{iter}\, U\, t\, u\, v)
\end{array}$$

Remark 9.15 (System **T** Recursor)**.** *It is of course possible to represent System* **T***'s* ***Recursor*** *(§8) in Church-style System* **F***, see e.g. [SU06, Prop. 11.3.6] or [GLT89, §11.5.1]. This however* ***cannot*** *be made in "*constant time*" (w.r.t. the number of β-reduction steps), see Rem. 9.32, §9.2.2 (see also [GLT89, §11.5.1]).*

However, the expressive power of System **F** *goes far beyond System* **T***, since the System* **F** *terms of type* $\mathtt{nat} \to \mathtt{nat}$ *are exactly those partial recursive functions which are provably total in Second-Order Arithmetic. See* **??** *below, as well as [SU06, §11 & §12] and [GLT89, §11 & §15].*

Example 9.16 (Lists)**.** *Given a type T, let*

$$\mathtt{list}(T) \quad := \quad (\forall X)\big(X \to (T \to X \to X) \to X\big)$$

(where $X \notin \mathrm{FV}(T)$). Given λ-terms $t_0, \dots, t_n, t, \ell, u, v$, let

$$\begin{array}{rcl}
[t_0; \dots; t_n]_T & := & \Lambda X.\ \lambda e : X.\ \lambda c : T \to X \to X.\ c\, t_0\,\big(\dots(c\, t_n\, e)\big) \\
\mathtt{nil}_T & := & \Lambda X.\ \lambda e : X.\ \lambda c : T \to X \to X.\ e \\
\mathtt{cons}_T\, t\, \ell & := & \Lambda X.\ \lambda e : X.\ \lambda c : T \to X \to X.\ c\, t\,(\ell\, X\, e\, c) \\
\mathtt{iter}_T\, U\, \ell\, u\, v & := & \ell\, U\, u\, v
\end{array}$$

so that

$$\frac{\mathcal{E} \vdash t_0 : T \quad \dots \quad \mathcal{E} \vdash t_n : T}{\mathcal{E} \vdash [t_0; \dots; t_n]_T : \mathtt{list}(T)} \qquad \frac{}{\mathcal{E} \vdash \mathtt{nil}_T : \mathtt{list}(T)} \qquad \frac{\mathcal{E} \vdash t : T \qquad \mathcal{E} \vdash \ell : \mathtt{list}(T)}{\mathcal{E} \vdash \mathtt{cons}_T\, t\, \ell : \mathtt{list}(T)}$$

$$\frac{\mathcal{E} \vdash \ell : \mathtt{list}(T) \qquad \mathcal{E} \vdash u : U \qquad \mathcal{E} \vdash v : T \to U \to U}{\mathcal{E} \vdash \mathtt{iter}_T\, U\, \ell\, u\, v : U}$$

Moreover,

$$\begin{array}{rcl}
\mathtt{cons}_T\,t\,\mathtt{nil}_T & \rhd^+_\beta & [t]_T \\
\mathtt{cons}_T\,t\,[t_0;\dots;t_n]_T & \rhd^+_\beta & [t;t_0;\dots;t_n]_T \\
\mathtt{iter}_T\,U\,\mathtt{nil}_T\,u\,v & \rhd^+_\beta & u \\
\mathtt{iter}_T\,U\,(\mathtt{cons}_T\,t\,\ell)\,u\,v & =_\beta & v\,t\,(\mathtt{iter}_T\,U\,\ell\,u\,v)
\end{array}$$

Remark 9.17 (Polymorphic Lists)**.** *Similarly as with products (Rem. 9.11) and sums (Rem. 9.13), we also have polymorphic versions of the operations of Ex. 9.16, namely:*

$$\begin{array}{rcl}
\mathtt{nil} & : & (\forall Y)\mathtt{list}(Y) \\
 & := & \Lambda Y.\,\mathtt{nil}_Y \\
 & & \\
\mathtt{cons} & : & (\forall Y)\big(Y \to \mathtt{list}(Y) \to \mathtt{list}(Y)\big) \\
 & := & \Lambda Y.\ \lambda x : Y.\ \lambda y : \mathtt{list}(Y).\ \mathtt{cons}_Y\,x\,y \\
 & & \\
\mathtt{iter} & : & (\forall Y)(\forall Z)\big(\mathtt{list}(Y) \to Z \to (Y \to Z \to Z) \to Z\big) \\
 & := & \Lambda Y.\ \Lambda Z.\ \lambda x : \mathtt{list}(Y).\ \lambda y : Z.\ \lambda z : Y \to Z \to Z.\ \mathtt{iter}_Y\,Z\,x\,y\,z
\end{array}$$

Exercise 9.18. *Give a polymorphic*

$$\mathtt{map} \quad : \quad (\forall X)(\forall Y)\big((X \to Y) \to \mathtt{list}(X) \to \mathtt{list}(Y)\big)$$

such that

$$\mathtt{map}\ T\ U\ f\ [t_0;\dots;t_n]_T \quad \rhd^+_\beta \quad [f t_0;\dots;f t_n]_U$$

Proof. Exercise! □

Remark 9.19 (Further Examples)**.** *Actually,* ***any*** *type of finite structures (together with its induction principle) specified by a finite first-order signature is representable in System* **F***, see [GLT89, §11.4]. See also [GLT89, §11.5.4] for a representation of (finite) trees whose branching type is a fixed arbitrary type T of System* **F***.*

9.1.4. Main Properties

The main property of System **F** is the strong normalization of typable terms. This non-trivial result, due to Girard, is proven in §9.3 (see [GLT89, §14]).

Theorem 9.20 (Strong Normalization (Girard))**.** *If $\mathcal{E} \vdash t : T$ then t is strongly normalizing for $\rhd_\beta$.*

As usual with non-pure normalizing typed λ-calculi, we obtain confluence via Newman's Lemma 3.35 (§3.4.3).

Theorem 9.21 (Confluence)**.** *For each typing context $\mathcal{E}$ and each type T, the relation $\rhd_\beta$ is confluent on $|\mathcal{E} \vdash T|$ (where $|\mathcal{E} \vdash T|$ is defined as in Rem. 4.11, §4.2.2).*

Explicit type annotations in terms make type-checking decidable in the following sense.

Remark 9.22 (Decidability of Typing)**.** *The following problem is decidable:*

- *Given a type T, a context $\mathcal{E}$ and a Church-style λ-term t, decide whether $\mathcal{E} \vdash t : T$ in Church-style System* **F***.*

Note that in contrast with Rem. 4.13 (§4.2.2, on the pure simply-typed λ-calculus), it is undecidable whether a given term is typable (in the sense of §4.2.1), see [SU06, §11.2].

A perhaps surprising but non-trivial fact is that ***type inhabitation*** *is undecidable for System* **F** *(see [SU06, §11.6]).*

We now turn to the analysis of typed normal forms. The analogue of Lem. 4.33 (§4.4.4) is the following.

Lemma 9.23 (Typed Normal Forms)**.** *If t is a λ-term in β-normal form and typable in the empty context, then t is of one of the following forms:*

$$\lambda x : T.u \qquad \text{or} \qquad \Lambda X.u$$

Lemma 9.23 is proven by a syntactic analysis similar to the proof of Lem. 4.33. Note that the term t is **not** assumed to be closed: the assumption that t is typable in the empty context implies that t contains no free λ-variable $x \in \mathcal{X}$, but t may contain free **type** variables $X \in \mathcal{V}_{\mathrm{Ty}}$.

The following Prop. 9.24 consists of (not all direct) consequences of Lem. 9.23. Actually, when it comes to the impredicative codings of §9.1.3, the analysis of typed normal forms in the empty context is more intricate than for the simply-typed versions (Lem. 4.33, §4.4.4). We come back on this in Rem. 9.38 (§9.2.3) below.

Proposition 9.24.

(1) There is no term t such that $\vdash t : \texttt{void}$.

(2) If $\vdash t : \texttt{unit}$ is a β-normal form then $t = \langle\rangle$.

(3) If $\vdash t : \texttt{bool}$ is a β-normal form then either $t = \texttt{true}$ or $t = \texttt{false}$.

(4) If $\vdash t : \texttt{nat}$ is a β-normal form then $t = \underline{n}$ for some $n \in \mathbb{N}$.

Proof. Exercise! □

9.2. Curry-Style System F

We now turn to **Curry-style** System **F**, which amounts to a polymorphic type system for the **pure** λ-terms of §3.2, namely

$$t, u \in \Lambda \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad tu$$

(where $x \in \mathcal{X}$). The types of Curry-style System **F** are the same as for the Church-style version, and the typing rules are given in Fig. 27.

$$(\textsc{Var})\ \frac{}{\mathcal{E} \vdash x : T}\ ((x : T) \in \mathcal{E})$$

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x.t : U \to T} \qquad (\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash t\,u : T}$$

$$(\forall^2\text{-I})\ \frac{\mathcal{E} \vdash t : T}{\mathcal{E} \vdash t : (\forall X)T}\ (X \notin \mathrm{FV}(\mathcal{E})) \qquad (\forall^2\text{-E})\ \frac{\mathcal{E} \vdash t : (\forall X)T}{\mathcal{E} \vdash t : T[U/X]}$$

Figure 27: Typing Rules of Curry-Style System **F**.

$$\begin{array}{rcl}
|x| & := & x \\
|\lambda x : T.t| & := & \lambda x.|t| \\
|t\,u| & := & |t|\,|u| \\
|\Lambda X.t| & := & |t| \\
|t\,T| & := & |t|
\end{array}$$

Figure 28: Erasure from Church-Style to Curry-Style System **F**.

Besides [SU06, §11.4], a good synthetic account on Curry-style System **F** is [Bar92, §4]. We insist on the (obvious) crucial difference with the Church-style version: Curry-style System **F** assigns types to **pure** λ-terms. As such, Curry-style System **F** turns out to be a particularly well suited setting for studying the computational behaviour of pure λ-terms, using polymorphic types as a structuring principle. This approach is developed in [Kri93, §8–§10].

Example 9.25. *Of course not every pure λ-term is typable in Curry-style System* **F**. *In particular, it follows from the Strong Normalization Theorem 9.35 (§9.2.3) that the term*

$$\Omega \quad = \quad (\underbrace{\lambda x.xx}_{\delta})\,\underbrace{\lambda x.xx}_{\delta}$$

*of Ex. 3.1 (§3.2.2) is **not** typable. But note that δ is typable as follows, where* `Id` $=$ $(\forall X)(X \to X)$ *(Ex. 9.6, §9.1.3):*

$$\dfrac{\dfrac{\dfrac{}{x : \mathtt{Id} \vdash x : (\forall X)(X \to X)}}{x : \mathtt{Id} \vdash x : \mathtt{Id} \to \mathtt{Id}} \qquad \dfrac{}{x : \mathtt{Id} \vdash x : \mathtt{Id}}}{\dfrac{x : \mathtt{Id} \vdash xx : \mathtt{Id}}{\vdash \lambda x.xx : \mathtt{Id} \to \mathtt{Id}}}$$

Further examples are given in §9.2.2, after having devised in §9.2.1 a type-preserving (and reflecting) erasure map from Church-style to Curry-style System **F**. The main properties of Curry-style System **F** are then stated in §9.2.3.

9.2.1. Erasing from Church-Style to Curry-Style System F

Figure 28 presents an erasure map from Church-style to Curry-style System **F**. This erasing preserves and reflects typability:

Proposition 9.26 (Typability)**.**

(1) If $\mathcal{E} \vdash t : T$ in Church-style System **F***, then $\mathcal{E} \vdash |t| : T$ in Curry-style System* **F***.*

(2) If $\mathcal{E} \vdash t : T$ in Curry-style System **F***, then there is a Church-style λ-term u such that $|u| = t$ and $\mathcal{E} \vdash u : T$ in Church-style System* **F***.*

9.2.2. Examples of Polymorphic Typings of Pure Lambda-Terms

It follows from §9.2.1 that the Curry-style System **F** has the same expressive power as the Church-style version. In particular, all examples for Church-style System **F** (§9.1.3) erase to examples for the Curry-style version. We just note here that in the relevant cases, this yields suitable polymorphic types for the **pure** λ-terms of Ex. 3.1 (§3.2.2).

Example 9.27 (Polymorphic Identity)**.** *Example 9.6 erases to*

$$\mathtt{id} \quad := \quad \lambda x.x \quad : \quad \underbrace{(\forall X)(X \to X)}_{\mathtt{Id}}$$

with

$$\mathtt{id}\, t \quad \rhd_\beta \quad t$$

Example 9.28 (Booleans)**.** *Example 9.9 erases to*

$$\frac{}{\mathcal{E} \vdash \mathtt{T} : \mathtt{bool}} \qquad \frac{}{\mathcal{E} \vdash \mathtt{F} : \mathtt{bool}} \qquad \frac{\mathcal{E} \vdash t : \mathtt{bool} \qquad \mathcal{E} \vdash u : T \qquad \mathcal{E} \vdash v : T}{\mathcal{E} \vdash \mathtt{if}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v : T}$$

where

$$\begin{array}{rcl} \mathtt{bool} & := & (\forall X)\big(X \to X \to X\big) \\ \mathtt{T} & := & \lambda x.\lambda y.x \\ \mathtt{F} & := & \lambda x.\lambda y.y \\ \mathtt{if}\ t\ \mathtt{then}\ u\ \mathtt{else}\ v & := & t\,u\,v \end{array}$$

Moreover, we have

$$\begin{array}{rcccl} \mathtt{if}\ \mathtt{T}\ \mathtt{then}\ u\ \mathtt{else}\ v & = & \big(\lambda x.\lambda y.x\big)\,u\,v & \rhd^+_\beta & u \\ \mathtt{if}\ \mathtt{F}\ \mathtt{then}\ u\ \mathtt{else}\ v & = & \big(\lambda x.\lambda y.y\big)\,u\,v & \rhd^+_\beta & v \end{array}$$

Example 9.29 (Product Types)**.** *Example 9.10 erases to*

$$\frac{\mathcal{E} \vdash t : T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash \mathtt{pair}\ t\,u : T \times U} \qquad \frac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_1 t : T} \qquad \frac{\mathcal{E} \vdash t : T \times U}{\mathcal{E} \vdash \pi_2 t : U}$$

where

$$\begin{array}{rcl} T \times U & := & (\forall X)\big((T \to U \to X) \to X\big) \\ \mathtt{pair}\ t\,u & := & \lambda p.p\,t\,u \\ \pi_1\, t & := & t\,(\lambda x.\lambda y.x) \\ \pi_2\, t & := & t\,(\lambda x.\lambda y.y) \end{array}$$

Moreover, we have

$$\begin{array}{rcccl} \pi_1\,(\mathtt{pair}\ t\,u) & = & \big(\lambda p.p\,t\,u\big)\,(\lambda x.\lambda y.x) & \rhd^+_\beta & t \\ \pi_2\,(\mathtt{pair}\ t\,u) & = & \big(\lambda p.p\,t\,u\big)\,(\lambda x.\lambda y.y) & \rhd^+_\beta & u \end{array}$$

Example 9.30 (Church's Numerals)**.** *Example 9.14 erases to*

$$\begin{array}{rcl}
\mathtt{nat} & := & (\forall X)\big(X \to (X \to X) \to X\big) \\
\underline{n} & := & \lambda z.\lambda s.s^n z \\
\mathsf{0} & := & \lambda z.\lambda s.z \\
\mathsf{S} & := & \lambda x.\lambda z.\lambda s.\ s\,(x\,z\,s) \\
\mathtt{iter}\,t\,u\,v & := & t\,u\,v
\end{array}$$

(where $n \in \mathbb{N}$). We have

$$\frac{}{\mathcal{E} \vdash \mathsf{0} : \mathtt{nat}} \qquad \frac{}{\mathcal{E} \vdash \mathsf{S} : \mathtt{nat} \to \mathtt{nat}} \qquad \frac{}{\mathcal{E} \vdash \underline{n} : \mathtt{nat}}\ (n \in \mathbb{N})$$

$$\frac{\mathcal{E} \vdash t : \mathtt{nat} \qquad \mathcal{E} \vdash u : U \qquad \mathcal{E} \vdash v : U \to U}{\mathcal{E} \vdash \mathtt{iter}\,t\,u\,v : U}$$

and

$$\begin{array}{rcl}
\mathsf{S}\,\underline{n} & \rhd^+_\beta & \underline{n+1} \\
\mathtt{iter}\,\mathsf{0}\,u\,v & \rhd^+_\beta & u \\
\mathtt{iter}\,(\mathsf{S}\,t)\,u\,v & =_\beta & v\,(\mathtt{iter}\,t\,u\,v)
\end{array}$$

Exercise 9.31. *Give terms*

$$\begin{array}{rcl}
\mathtt{add} & : & \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \\
\mathtt{mult} & : & \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat} \\
\mathtt{exp} & : & \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat}
\end{array}$$

such that

$$\begin{array}{rcl}
\mathtt{add}\,\underline{n}\,\underline{m} & \rhd^*_\beta & \underline{n+m} \\
\mathtt{mult}\,\underline{n}\,\underline{m} & \rhd^*_\beta & \underline{n \times m} \\
\mathtt{exp}\,\underline{n}\,\underline{m} & \rhd^*_\beta & \underline{n^m}
\end{array}$$

PROOF. Exercise! □

Remark 9.32 (System **T** Recursor)**.** *Remark 9.15 (§9.1.3) implies that it is of course possible to represent System* **T***'s* ***Recursor*** *(§8) in Curry-style System* **F** *(see e.g. [SU06, Prop. 11.3.6]). This however* ***cannot*** *be made in constant time ([Par89]), in the sense that there is no (pure) λ-term* Rec *such that there is $k \in \mathbb{N}$ such that for all λ-terms $t, u, v \in \Lambda$ we have*

$$\mathtt{Rec}\,u\,v\,\mathsf{0} \ \rhd^k_\beta\ u \qquad \text{and} \qquad \mathtt{Rec}\,u\,v\,(\mathsf{S}\,t) \ \rhd^k_\beta\ v\,t\,(\mathtt{Rec}\,u\,v\,t)$$

Remark 9.33 (Polymorphic Versions)**.** *Similarly as in Rem. 9.11 (§9.1.3) we also have* ***polymorphic*** *versions of Ex. 9.28, Ex. 9.29 and Ex. 9.30. Namely*

$$\begin{array}{rclcl}
\lambda x.\lambda y.\ \mathtt{pair}\,x\,y & = & \lambda x.\lambda y.\lambda p.\ pxy & : & (\forall XY)\big(X \to Y \to X \times Y\big) \\
\lambda p.\ \pi_i x & = & \lambda p.\ p(\lambda x_1.\lambda x_2.\ x_i) & : & (\forall X_1 X_2)\big(X_1 \times X_2 \to X_i\big)
\end{array}$$

and

$$\begin{array}{rclcl}
\lambda b.\lambda x.\lambda y.\ \mathtt{if}\ b\ \mathtt{then}\ x\ \mathtt{else}\ y & = & \lambda b.\lambda x.\lambda y.\ b\,x\,y & : & (\forall X)\big(\mathtt{bool} \to X \to X \to X\big) \\
\lambda x.\lambda y.\lambda z.\ \mathtt{iter}\,x\,y\,z & = & \lambda x.\lambda y.\lambda z.\ x\,y\,z & : & (\forall X)\big(\mathtt{nat} \to X \to (X \to X) \to X\big)
\end{array}$$

9.2.3. Main Properties

A striking aspect of Curry-style System **F** is that a given term can be given many types. For instance (with the notations of Ex. 9.27, §9.2.2):

$$\begin{array}{lcl}
\texttt{id} & : & \texttt{Id} \\
\texttt{id} & : & \texttt{Id} \to \texttt{Id} \\
\texttt{id} & : & (\texttt{Id} \to \texttt{Id}) \to (\texttt{Id} \to \texttt{Id}) \\
\texttt{id} & : & \cdots
\end{array}$$

This cannot happen in the **Church-style** version, since typing of Church-style λ-terms is **syntax-directed**: there is at most one typing derivation for given term (in a given typing environment).

In contrast, typing in **Curry-style** System **F** is not syntax-directed. This fact (which may be seen as an actual **feature** of the Curry-style version, as in *e.g.* [Kri93, §8–§10]) makes typing somehow more difficult with the Curry-style version than with the Church-style one. A typical instance of this is Subject Reduction, for which we refer to [SU06, §11.4] (see also [Kri93, §8.2] or [Bar92, §4.2]).

Theorem 9.34 (Subject Reduction)**.** *In Curry-style System* **F***, if* $\mathcal{E} \vdash t : T$ *and* $t \rhd_\beta u$ *then* $\mathcal{E} \vdash u : T$*.*

Typed Curry-style terms are strongly normalizing.

Theorem 9.35 (Strong Normalization)**.** *In Curry-style System* **F***, if* $\mathcal{E} \vdash t : T$ *then* t *is strongly normalizing for* $\rhd_\beta$*.*

We prove Thm. 9.35 in §9.3. We shall actually deduce the Strong Normalization for the Church-style version (Thm. 9.20, §9.1.4) from Thm. 9.35.

Remark 9.36 (Undecidability of Typing)**.** *The following problem is* ***undecidable****:*

- *Given a type* T*, a context* $\mathcal{E}$ *and a* λ*-term* t*, decide whether* $\mathcal{E} \vdash t : T$ *in Curry-style System* **F***.*

This result, which is not as easy as it may seems, is due to Wells ([SU06, Thm. 11.4.7]).

The analogue of Prop. 9.24 (§9.1.4) for Curry-style terms is the following (with the notations of §9.2.2), which can be proven from Prop. 9.24 via the erasure map of §9.2.1.

Proposition 9.37.

(1) There is no term t *such that* $\vdash t : \texttt{void}$*.*

(2) If $\vdash t : \texttt{unit}$ *is a* β*-normal form then* $t = \texttt{id}$*.*

(3) If $\vdash t : \texttt{bool}$ *is a* β*-normal form then either* $t = \texttt{T}$ *or* $t = \texttt{F}$*.*

(4) If $\vdash t : \texttt{nat}$ *is a* β*-normal form then* $t = \underline{n}$ *for some* $n \in \mathbb{N}$*.*

Remark 9.38 (Analysis of Closed Typed Normal Forms)**.** *Note that neither Prop. 9.37 nor Prop. 9.24 (§9.1.4) deal with the impredicative coding of product types (nor with that of sum types).*

As alluded to before Prop. 9.24 (§9.1.4), regarding the impredicative codings of §9.1.3 and §9.2.2, the analysis of closed typed normal forms of System **F** *is more intricate than in the simply typed case.*

*We refer to [Kri93, §8.4] for a general theory of "*data-types*" which makes it possible to obtain the expected results in **Curry-style** System* **F**. *For instance, under suitable assumptions on U and V, the machinery of [Kri93, §8.4] gives the following (with the notation of §9.2.2):*

- *If $\vdash t : U \times V$ is a β-normal form then $t = \mathtt{pair}\, u\, v$ with $\vdash u : U$ and $\vdash v : V$.*

9.3. Strong Normalization

We prove here that the typed terms of Church-style and Curry-style System **F** are strongly normalizing. We follow the same general outline as [SU06, §11.5]. Other important references are [Kri93, §8.3] (as well as [Bar92, §4.3]), and of course [GLT89, §14].

Theorem 9.39 (Girard)**.**

(1) If t is typable term of Church-style System **F**, *then t is strongly normalizing.*

(2) If t a pure λ-term typable in Curry-style System **F**, *then t is strongly normalizing.*

Theorem 9.39.(1) follows from Thm. 9.39.(2) using a combinatorial argument (actually formalizable in Second-Order Arithmetic §**??**) discussed in §9.3.1. The core of Thm. 9.39, namely Thm. 9.39.(2), is proven in §9.3.2 below, using an extension of the **reducibility method** of §5.

9.3.1. Erasing from Church-style to Curry-style System F

We show here Thm. 9.39.(1) assuming Thm. 9.39.(2): If Curry-style System **F** is strongly normalizing, then so is Church-style System **F**.

The proof is based on the erasure $|-|$ of Fig. 27 (§9.2.1). Recall from Prop. 9.26 that $|-|$ preserves and reflects typability.

Erasing almost preserves reductions. In order to formulate this, we need to split the relation $\rhd_\beta$ of β-reduction for **Church-style** System **F** as

$$\rhd_\beta \quad = \quad \rhd_\lambda \cup \rhd_\Lambda$$

where $\rhd_\lambda$ (resp. $\rhd_\Lambda$) is the closure under the **Congruence Rules** of Fig. 25 (§9.1.1) of

$$(\lambda x : T.t)u \quad \rhd_\lambda \quad t[u/x] \qquad \text{resp.} \qquad (\Lambda X.t)T \quad \rhd_\Lambda \quad t[T/X]$$

Lemma 9.40 (Reduction and Erasing)**.** *Let t, u be (possibly untyped) terms of **Church-style** System* **F**. *Then,* *(1) if $t \rhd_\lambda u$ then $|t| \rhd_\beta |u|$,*

(2) if $t \rhd_\Lambda u$ then $|t| = |u|$.

Remark 9.41. *Note that the strong normalization of $\rhd_\Lambda$ is trivial (even on non typable terms) since the number of $\rhd_\Lambda$-redexes strictly decreases at each $\rhd_\Lambda$-step.*

Proposition 9.42. *If Curry-style System **F** is strongly normalizing then Church-style System **F** is also strongly normalizing.*

Proof. Exercise! □

9.3.2. Strong Normalization of Curry-Style System F

We prove here Thm. 9.39.(2). We follow the usual (and only known) method of Girard's **reducibility candidates**. The general pattern of the method is that of §5 for normalization of the simply-typed λ-calculus, extended in §8.2.2 and §8.2.5 to System **T**, and similar to the (modified) realizability interpretation of §8.3.

Our argument actually follows the lines of [Kri93, §8.3] (see also [SU06, §11.5] or [Bar92, §4.3]), which is somehow simpler than [GLT89, §14] (as the latter directly proves the strong normalization of **Church-style** System **F**).

Notation 9.43. *For the remaining of this §9.3, we let $\mathcal{SN}$ stand for strong normalization w.r.t. β-reduction (see Notation 5.24, §5.5.1).*

Roughly speaking, the basic idea is the following. Assume that, along the lines of §5, we want to define an interpretation $[\![T]\!]$ for each type T of System **F**. For arrow types $U \to T$, we can proceed as in §5, and set

$$[\![U \to T]\!] \quad := \quad [\![U]\!] \Rrightarrow [\![T]\!]$$

(where $\Rrightarrow$ is as in Def. 5.8, §5.3.2).

However, we have a problem in the case of a type quantification $(\forall X)T$. A tentative definition of $[\![(\forall X)T]\!]$ could be to take the set of all terms $t \in \mathcal{SN}$ such that

$$\text{for all type } U,\ t \in [\![T[U/X]]\!]$$

But this definition is ill-founded since $T[U/X]$ can be $(\forall X)T$ itself! For instance, the above scheme would lead for $[\![(\forall X)X]\!]$ to take the set of all $t \in \mathcal{SN}$ such that

$$\text{for all type } T,\ t \in [\![T]\!]$$

The solution found by Girard is to define **a priori** a suitable family of sets $\mathcal{SAT} \subseteq \mathcal{P}(\mathcal{SN})$, and **then** to interpret type quantifications $(\forall X)T$ by universal quantifications over $S \in \mathcal{SAT}$. For instance, in the case of $(\forall X)T$ this leads to take

$$[\![(\forall X)X]\!] \quad = \quad \bigcap \mathcal{SAT}$$

Technically, a major difference with the setting of §5 is that type interpretations $[\![T]\!]$ should now be parametrized by **type valuations** $\rho : \mathcal{V}_{\mathrm{Ty}} \to \mathcal{SAT}$.

9.3.3. Reducibility Candidates

We thus arrive to the (then) easy adaptation of §5.5 to Curry-style System **F**. Consider the following simple notion of **elimination context**:

$$E[\,] \in \mathrm{Elim} \quad ::= \quad [\,] \quad | \quad E[\,]\,t$$

Similarly as in Notation 5.25 (§5.5.2), write $(t_1, \dots, t_n) \rhd_\beta (u_1, \dots, u_n)$ when there is some $i \in \{1, \dots, n\}$ such that $t_i \rhd_\beta u_i$ and $t_j = u_j$ for all $j \neq i$.

The key Lem. 5.26 and Lem. 5.27 specialize to the following. We refer to §5.5.2 for the proof.

Lemma 9.44. *On pure λ-terms we have the following.*

(Weak Standardization) *If $E[(\lambda x.t)u] \rhd_\beta v$, then either $v = E[t[u/x]]$ or v is of the form $E'[(\lambda x.t')u']$ with $(E[\,], t, u) \rhd_\beta (E'[\,], t', u')$.*

(Weak Head Expansion) *If $E[t[u/x]] \in \mathcal{SN}$ and $u \in \mathcal{SN}$ then $E[(\lambda x.t)u] \in \mathcal{SN}$.*

The notion of Saturated Set (Def. 5.28, §5.5.3) for **pure** λ-terms simplifies to the following.

Definition 9.45 (Saturated Sets)**.** *The set $\mathcal{SAT}$ of **saturated sets** is the set of all $A \subseteq \mathcal{SN}$ such that*

(i) $E[x] \in A$ whenever $E[\,] \in \mathrm{Elim} \cap \mathcal{SN}$, and

(ii) $E[(\lambda x.t)u] \in A$ whenever $u \in \mathcal{SN}$ and $E[t[u/x]] \in A$.

Let us first check that $\mathcal{SAT}$ is not empty.

Lemma 9.46. $\mathcal{SN} \in \mathcal{SAT}$

Proof. Exercise! □

Note also that thanks to the first clause of $\mathcal{SAT}$, saturated sets are not empty, since $S \in \mathcal{SAT}$ implies $x \in S$ for every variable $x \in \mathcal{X}$.

We shall interpret System **F** types by saturated sets. In order to achieve this, we need the following closure properties of $\mathcal{SAT}$.

Lemma 9.47.

(1) Given $A, B \in \mathcal{SAT}$, we have $A \sqsupset\!\!\!\rightarrow B \in \mathcal{SAT}$ (where $\sqsupset\!\!\!\rightarrow$ is as in Def. 5.8, §5.3.2).

(2) Given $\mathcal{S} \subseteq \mathcal{SAT}$, we have $\bigcap \mathcal{S} \in \mathcal{SAT}$.

We can finally define the type interpretation. Given $\rho : \mathcal{V}_{\mathrm{Ty}} \to \mathcal{SAT}$, define $[\![T]\!]\rho$ by induction on T as follows:

$$\begin{array}{rcl} [\![X]\!]\rho & := & \rho(X) \\ [\![U \to T]\!]\rho & := & [\![U]\!]\rho \sqsupset\!\!\!\rightarrow [\![T]\!]\rho \\ [\![(\forall X)T]\!]\rho & := & \bigcap_{S \in \mathcal{SAT}} [\![T]\!]\rho[S/X] \end{array}$$

Lemma 9.48. *For all type T and all $\rho : \mathcal{V}_{\mathrm{Ty}} \to \mathcal{SAT}$ we have $[\![T]\!]\rho \in \mathcal{SAT}$.*

9.3.4. Adequacy

We now turn to the **adequacy** of our type interpretation. Recall from Def. 5.7 (§5.3.2) that this amounts to the following:

- If $x_1 : U_1, \dots, x_n : U_n \vdash t : T$ is derivable in Curry-style System **F**, then for all $\rho : \mathcal{V}_{\mathrm{Ty}} \to \mathcal{SAT}$ and all $u_1 \in [\![U_1]\!]\rho, \dots, u_n \in [\![U_n]\!]\rho$ we have $t[\vec{u}/\vec{x}] \in [\![T]\!]\rho$.

We need the following "semantic" Substitution Lemma.

Lemma 9.49 (Substitution)**.** *Given T, U and $\rho : \mathcal{V}_{\mathrm{Ty}} \to \mathcal{SAT}$, we have $[\![T[U/X]]\!]\rho = [\![T]\!]\rho[[\![U]\!]\rho/X]$.*

We now prove the adequacy of the interpretation. We reason by induction on typing derivations and by cases on the last applied rule. In the following, we assume that the contexts $\mathcal{E}$ are of the form $x_1 : U_1, \dots, x_n : U_n$.

Case of

$$(\textsc{Var})\ \frac{(x : T) \in \mathcal{E}}{\mathcal{E} \vdash x : T}$$

Trivial.

Case of

$$(\to\text{-I})\ \frac{\mathcal{E}, x : U \vdash t : T}{\mathcal{E} \vdash \lambda x.t : U \to T}$$

Let $\vec{u} \in [\![\vec{U}]\!]\rho$ and $u \in [\![U]\!]\rho$. We have to show $(\lambda x.t[\vec{u}/\vec{x}])u \in [\![T]\!]\rho$.

By induction hypothesis we have $t[\vec{u}/\vec{x}, u/x] \in [\![T]\!]\rho$, and we are done since $[\![T]\!]\rho \in \mathcal{SAT}$ and $u \in \mathcal{SN}$.

Case of

$$(\to\text{-E})\ \frac{\mathcal{E} \vdash t : U \to T \qquad \mathcal{E} \vdash u : U}{\mathcal{E} \vdash tu : T}$$

By definition of $(-) \mathrel{\square\!\!\to} (-)$.

Case of

$$(\forall^2\text{-I})\ \frac{\mathcal{E} \vdash t : T}{\mathcal{E} \vdash t : (\forall X)T}\ (X \notin \mathrm{FV}(\mathcal{E}))$$

Let $\vec{u} \in [\![\vec{U}]\!]\rho$ and $S \in \mathcal{SAT}$. We have to show $t[\vec{u}/\vec{x}] \in [\![T]\!]\rho[S/X]$.

Note that $X \notin \mathrm{FV}(\vec{U})$, so that $\vec{u} \in [\![\vec{U}]\!]\rho[S/X]$, and we are done by induction hypothesis.

Case of

$$(\forall^2\text{-E})\ \frac{\mathcal{E} \vdash t : (\forall X)T}{\mathcal{E} \vdash t : T[U/X]}$$

Let $\vec{u} \in [\![\vec{U}]\!]\rho$. By induction hypothesis we have $t[\vec{u}/\vec{x}] \in [\![T]\!]\rho[[\![U]\!]\rho/X]$ since $[\![U]\!]\rho \in \mathcal{SAT}$, and we are done by Lem. 9.49. □

We then directly obtain the Strong Normalization Theorem.

Theorem 9.50 (Thm. 9.39.(2))**.** *If $\mathcal{E} \vdash t : T$ in Curry-style System* **F** *then $t \in \mathcal{SN}$.*

Proof. Exercise! $\square$

References

[AC98] R. M. Amadio and P.-L. Curien. *Domains and Lambda-Calculi*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998. 7, 23, 37, 41, 43, 58, 109

[Awo10] S. Awodey. *Category Theory*. Oxford University Press, Inc., USA, 2nd edition, 2010. 57

[Bar84] H.P. Barendregt. *The Lambda-Calculus, its Syntax and Semantics*. Studies in Logic and the Foundation of Mathematics. North Holland, 1984. Second edition. 23, 31, 32, 37, 38, 39, 40, 80

[Bar92] H.P. Barendregt. Lambda Calculi with Types. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2. Oxford University Press, 1992. 7, 23, 31, 35, 41, 58, 86, 139, 147, 150, 151, 152

[BBC98] S. Berardi, M. Bezem, and T. Coquand. On the Computational Content of the Axiom of Choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998. 82

[BBJ07] G. S. Boolos, J. P. Burgess, and R. C. Jeffrey. *Computability and Logic*. Cambridge University Press, fifth edition, 2007. 80, 101, 102, 103, 108

[Bee85] M. J. Beeson. *Foundations of Constructive Mathematics*. Ergebnisse der Mathematik und ihrer Grenzgebiete. Springer, Berlin, Heidelberg, 1985. 7, 98, 101, 114, 131

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998. 29, 30, 33

[BO05] U. Berger and P. Oliva. Modified bar recursion and classical dependent choice. *Lecture Notes in Logic*, 20:89–107, 2005. 82

[Bus98a] S. R. Buss. An Introduction to Proof Theory. In S. R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 1–78. Elsevier, Amsterdam, 1998. 10, 16, 71, 83, 91, 92, 94, 95, 160, 162, 163, 164

[Bus98b] S. R. Buss. First-Order Proof Theory of Arithmetic. In S. R. Buss, editor, *Handbook of Proof Theory*, volume 137 of *Studies in Logic and the Foundations of Mathematics*, pages 79–147. Elsevier, Amsterdam, 1998. 97

[Cha12] A. Charguéraud. The Locally Nameless Representation. *Journal of Automated Reasoning*, 49(3):363–408, 2012. 27

[DP02] B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order*. CUP, 2nd edition, 2002. 163

[Fri78] Harvey Friedman. Classically and intuitionistically provably recursive functions. In Gert H. Müller and Dana S. Scott, editors, *Higher Set Theory*, pages 21–27, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg. 114

[Gal95] J. H. Gallier. On the Correspondance Between Proofs and λ-Terms. In P. de Groote, editor, *Cahiers du Centre de Logique*. Université Catholique de Louvain, 1995. 41, 47, 49, 57, 83, 95

[Gir91] J.-Y. Girard. A new constructive logic: Classical logic. *Math. Struct. Comput. Sci.*, 1(3):255–296, 1991. 22

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989. 10, 16, 41, 58, 98, 101, 122, 125, 126, 138, 139, 141, 144, 145, 151, 152, 160

[Hof97] M. Hofmann. Syntax and semantics of dependent types. In Andrew M. Pitts and P.Editors Dybjer, editors, *Semantics and Logics of Computation*, Publications of the Newton Institute, page 79–130. Cambridge University Press, 1997. 70

[Jac01] B. Jacobs. *Categorical Logic and Type Theory*. Studies in logic and the foundations of mathematics. Elsevier, 2001. 70

[Joh82] P.T. Johnstone. *Stone Spaces*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1982. 164

[Koh08] U. Kohlenbach. *Applied Proof Theory: Proof Interpretations and their Use in Mathematics*. Springer Monographs in Mathematics. Springer, 2008. 7, 17, 82, 100, 106, 109, 110, 114, 116, 122, 131, 132, 133, 137

[Kri93] J.-L. Krivine. *Lambda-Calculus, Types and Models*. Ellis Horwood, 1993. 23, 27, 37, 58, 131, 139, 147, 150, 151, 152

[Kri01] J.-L. Krivine. Typed lambda-calculus in classical Zermelo-Frænkel set theory. *Arch. Math. Log.*, 40(3):189–205, 2001. 58, 131

[Kri09] J.-L. Krivine. Realizability in classical logic. In *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et synthèses*, pages 197–229. Société Mathématique de France, 2009. 19, 58, 131

[Kri12] J.-L. Krivine. Realizability algebras II : new models of ZF + DC. *Log. Methods Comput. Sci.*, 8(1), 2012. 132

[KV65] S. C. Kleene and R. E. Vesley. *The Foundations of Intuitionistic Mathematics*, volume 39 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1965. 131

[Lev03] P. B. Levy. *Call-By-Push-Value*. Semantics Structures in Computation. Springer, Dordrecht, 2003. 44

[LS86] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic.* CUP, 1986. 57, 58

[Mel09] P.-A. Melliès. Categorical semantics of linear logic. In *Interactive models of computation and program behaviour*, volume 27 of *Panoramas et Synthèses.* SMF, 2009. 16, 160

[Mel17] P.-A. Melliès. *Une étude micrologique de la négation.* Habilitation à diriger des recherches, Université Paris Diderot, 2017. 22

[Miq11] A. Miquel. Existential witness extraction in classical realizability and via a negative translation. *Logical Methods in Computer Science*, 7(2), 2011. 116

[Odi99] P. Odifreddi. *Classical Recursion Theory*, volume 143 of *Studies in Logic and the Foundations of Mathematics.* Elsevier, second edition, 1999. 101, 109

[Par89] M. Parigot. On the Representation of Data in Lambda-Calculus. In *Proceedings of CSL'89*, volume 440 of *LNCS*, pages 309–321, 1989. 149

[Pie02] B. C. Pierce. *Types and Programming Languages.* The MIT Press, 1st edition, 2002. 7, 41, 47, 48, 58, 138, 141

[Run05] V. Runde. *A Taste of Topology.* Universitext. Springer New York, 2005. 164

[Sel01] P. Selinger. Control Categories and Duality: on the Categorical Semantics of the Lambda-Mu Calculus. *Mathematical Structures in Computer Science*, 11:207–260, 2001. 19, 58

[Sim10] S. G. Simpson. *Subsystems of Second Order Arithmetic.* Perspectives in Logic. Cambridge University Press, 2nd edition, 2010. 98, 112, 113, 163

[SU06] M. H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics.* Elsevier Science Inc., 2006. 1, 7, 8, 10, 11, 12, 13, 14, 16, 17, 19, 20, 23, 27, 34, 35, 36, 37, 41, 43, 44, 46, 47, 52, 56, 57, 58, 70, 71, 78, 80, 81, 82, 83, 86, 95, 99, 100, 101, 102, 106, 107, 108, 110, 114, 115, 116, 122, 124, 125, 126, 131, 132, 137, 138, 139, 141, 144, 146, 147, 149, 150, 151, 152, 160, 162

[Tro73] A. S. Troelstra. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *LNM.* Springer Verlag, 1973. 100, 110, 114, 122, 124, 126, 129, 131, 132

[TS00] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 2nd edition, 2000. 47, 49, 83, 98, 101

[TvD88a] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Volume 1*, volume 121 of *Studies in Logic and the Foundations of Mathematics.* Elsevier,

1988. 7, 10, 11, 12, 17, 19, 41, 58, 67, 79, 80, 82, 83, 99, 100, 101, 102, 106, 110, 112, 114, 115, 116, 131, 137

[TvD88b] A. S. Troelstra and D. van Dalen. *Constructivism in Mathematics, Volume 2*, volume 123 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1988. 7, 11, 83, 95, 100, 114, 122

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `https://homotopytypetheory.org/book`, Institute for Advanced Study, 2013. 70, 86

[vD04] D. van Dalen. *Logic and Structure*. Universitext. Springer, fourth edition, 2004. 7, 11, 16, 17, 19, 20, 47, 71, 76, 78, 80, 82, 83, 91, 92, 93, 99, 101, 102, 106, 107, 108, 109, 113, 126, 127, 128, 162, 163

[Wil70] S. Willard. *General Topology*. Addison-Wesley, 1970. 164

A. Completeness of Classical Propositional Logic

This Appendix discusses the completeness of the (classical) system NK_0 (Thm. 2.30, §2.3). See §A.4 for a comparison with other usual statements.

A.1. A Semantically Reversible Sequent Calculus

In this §A.1 we show the completeness of a Gentzen-style **sequent calculus** for a fragment of NK_0. We consider (for simplicity) formulae on the grammar:

$$A, B \quad ::= \quad \mathsf{p} \quad | \quad A \Rightarrow B \quad | \quad \bot$$

where p is an atomic proposition. Beware that in contrast with minimal logic (§2.4), we assume the proposition $\bot$. See [SU06, §7] (also [GLT89, §5], [Bus98a, §1.2] or [Mel09, §1]) for more on Gentzen-style sequent calculi.

Definition A.1.

*(1) A **sequent** is a pair $\Delta \vdash \Gamma$, where Δ and Γ are (possibly empty) **lists** of formulae.*

*(2) A sequent $A_1, \dots, A_n \vdash B_1, \dots, B_m$ is **valid** if $(A_1 \wedge \cdots \wedge A_n) \Rightarrow (B_1 \vee \cdots \vee B_m)$ is valid (in the sense of Def. 2.10, §2.1.1).*

(3) A rule

$$\frac{\Delta_1 \vdash \Gamma_1 \quad \dots \quad \Delta_n \vdash \Gamma_n}{\Delta \vdash \Gamma}$$

is

- ***valid** if the sequent $\Delta \vdash \Gamma$ is valid whenever the sequents $\Delta_1 \vdash \Gamma_1, \dots, \Delta_n \vdash \Gamma_n$ are all valid;*
- ***semantically reversible** if each sequent $\Delta_i \vdash \Gamma_i$ is valid whenever the sequent $\Delta \vdash \Gamma$ is valid.*

Remark A.2. *Consider a sequent $A_1, \dots, A_n \vdash B_1, \dots, B_m$. If $n = 0$, this sequent is valid in the sense of Def. A.1.(2) if and only if the formula $\top \Rightarrow (B_1 \vee \cdots \vee B_m)$ is valid, that is if and only if $B_1 \vee \cdots \vee B_m$ is valid. Symmetrically, if $m = 0$ the above sequent is valid if and only if $(A_1 \wedge \cdots \wedge A_n) \Rightarrow \bot$ (that is $\neg(A_1 \wedge \cdots \wedge A_n)$) is valid. As a consequence, the validity of the empty sequent $\vdash$ (i.e. with $n = m = 0$ above) is equivalent to the validity of $\top \Rightarrow \bot$. It follows that the empty sequent $\vdash$ is **not** valid.*

We shall consider deduction with the rules of Fig. 29.

Remark A.3. *All rules of Fig. 29 are valid.*

Lemma A.4. *All rules of Fig. 29 are semantically reversible.*

Proof. Exercise! $\square$

$$\frac{}{\Delta, A \vdash A, \Gamma} \qquad \frac{\Delta \vdash \Gamma}{\varsigma(\Delta) \vdash \varsigma(\Gamma)} \qquad \frac{}{\Delta, \bot \vdash \Gamma} \qquad \frac{\Delta \vdash \Gamma}{\Delta \vdash \bot, \Gamma}$$

$$\frac{\Delta \vdash A, \Gamma \qquad \Delta, B \vdash \Gamma}{\Delta, A \Rightarrow B \vdash \Gamma} \qquad \frac{\Delta, A \vdash B, \Gamma}{\Delta \vdash A \Rightarrow B, \Gamma}$$

Figure 29: A Reversible Sequent Calculus (where ς is a permutation).

Theorem A.5 (Completeness)**.** *If $\Delta \vdash \Gamma$ is a valid sequent, then $\Delta \vdash \Gamma$ is derivable with the rules of Fig. 29.*

Theorem A.5 can be proved by iteratively applying Lem. A.4 along an induction on the following measure $|\Delta \vdash \Gamma|$ of sequents $\Delta \vdash \Gamma$. First, $|A|$ is defined by induction on A as

$$\begin{array}{rcl} |p| & := & 0 \\ |\bot| & := & 1 \\ |A \Rightarrow B| & := & |A| + |B| + 1 \end{array}$$

Then let $|A_1, \dots, A_n \vdash B_1, \dots, B_m| := |A_1| + \cdots + |A_n| + |B_1| + \cdots + |B_m|$.

Proof. Exercise! □

Remark A.6. *We stated Thm. A.5 for the logical connectives $\Rightarrow$ and $\bot$ only because this is sufficient for the completeness of NK_0, as the other connectives $(\land, \lor, \top)$ are **classically** definable from $\Rightarrow, \bot$ (see Lem. A.10, §A.3 below). But Thm. A.5 actually extends to the full language of NK_0, using e.g. the following semantically reversible rules:*

$$\frac{\Delta \vdash \Gamma}{\Delta, \top \vdash \Gamma} \qquad \frac{}{\Delta \vdash \top, \Gamma}$$

$$\frac{\Delta, A, B \vdash \Gamma}{\Delta, A \land B \vdash \Gamma} \qquad \frac{\Delta \vdash A, \Gamma \qquad \Delta \vdash B, \Gamma}{\Delta \vdash A \land B, \Gamma}$$

$$\frac{\Delta, A \vdash \Gamma \qquad \Delta, B \vdash \Gamma}{\Delta, A \lor B \vdash \Gamma} \qquad \frac{\Delta \vdash A, B, \Gamma}{\Delta \vdash A \lor B, \Gamma}$$

A.2. Translation to NK_0

We now consider a translation from the classical sequent calculus of Fig. 29 to NK_0.

Proposition A.7. *If $A_1, \dots, A_n \vdash B_1, \dots, B_m$ is derivable using the rules of Fig. 29 then $A_1, \dots, A_n, \neg B_1, \dots, \neg B_m \vdash \bot$ is derivable in NK_0.*

Proof. Exercise! □

Remark A.8. *Proposition A.7 extends to the full language of NK_0 if one extends the sequent calculus of Fig. 29 with the rules of Rem. A.6.*

A.3. Completeness of NK_0

We finally show the completeness of NK_0.

Theorem A.9 (Completeness of NK_0 (Thm. 2.30, §2.3))**.** *If $\Delta \vdash A$ is a valid sequent (in the language of §2), then it is provable in NK_0.*

We obtain Thm. A.9 from Thm. A.5 and Prop. A.7 via the following, which says that all the formulae of §2 can be defined (**w.r.t. classical logic**) in the language of §A.1.

Lemma A.10. NK_0 *proves the following:*

(1) $\vdash \top \Leftrightarrow (\bot \Rightarrow \bot)$

(2) $\vdash (A \wedge B) \Leftrightarrow ((A \Rightarrow B \Rightarrow \bot) \Rightarrow \bot)$

(3) $\vdash (A \vee B) \Leftrightarrow ((A \Rightarrow \bot) \Rightarrow (B \Rightarrow \bot) \Rightarrow \bot)$

Proof. Exercise! $\square$

Proof of Theorem A.9.

Proof. Let $\Delta \vdash A$ be a sequent in the language of §2. Assume that $\Delta \vdash A$ is valid. First, using Lem. A.10 (and an induction on formulae) we can put $\Delta \vdash A$ in the language of §A.1. Hence Thm. A.5 implies that $\Delta \vdash A$ is provable in the sequent calculus of Fig. 29. Then Prop. A.7 implies that NJ_0 proves $\Delta \vdash \neg\neg A$, and we conclude with (DNE) (Cor. 2.23, §2.2.1). $\square$

A.4. Comparison with other (Compactness and) Completeness Statements

Theorem A.9 is essentially equivalent to [SU06, Thm. 6.1.10] (see also [Bus98a, 1.1.3.(1) & 1.1.4]). A common stronger formulation of completeness for classical propositional logic is *e.g.* [vD04, Thm. 1.5.13] (see also [Bus98a, 1.1.3.(2)]). In order to state it, we first need to make the following definition, based on [vD04, Def. 1.4.2 & Def. 1.5.2].

Definition A.11. *Let $\mathbf{\Phi}$ be an* ***arbitrary set*** *of formulae.*

(1) We write $\mathbf{\Phi} \vdash A$ if there are formulae $A_1, \dots, A_n \in \mathbf{\Phi}$ such that $A_1, \dots, A_n \vdash A$ is derivable in NK_0.

(2) We say that $\mathbf{\Phi}$ is ***inconsistent*** *if $\mathbf{\Phi} \vdash \bot$.*

(3) We say that $\mathbf{\Phi}$ is ***consistent*** *if it is* ***not*** *inconsistent.*

Hence a set $\mathbf{\Phi}$ of formulae is consistent iff $\mathbf{\Phi} \nvdash \bot$.

Definition A.12. *Let $\mathbf{\Phi}$ be an* ***arbitrary set*** *of formulae.*

(1) Given a valuation v, we write $v \models \mathbf{\Phi}$ if $v \models B$ for all $B \in \mathbf{\Phi}$.

(2) Given a formula A, we write $\mathbf{\Phi} \models A$ when for every valuation v, we have $v \models A$ whenever $v \models \mathbf{\Phi}$.

We can now state [vD04, Thm. 1.5.13] (see also [Bus98a, 1.1.3.(2)]).

Theorem A.13. *If $\mathbf{\Phi}$ is consistent, then there is a valuation v such that $v \models \mathbf{\Phi}$.*

Note that Thm. A.13 easily gives Thm A.9.

Proof. Assume that $\Delta \vdash A$ is valid. Let $\mathbf{\Phi}$ be the set of formulae occurring in Δ. Then $\mathbf{\Phi} \cup \{\neg A\}$ is inconsistent since if it were consistent, Thm. A.13 would imply that there is a valuation v such that $v \not\models A$ and $v \models \mathbf{\Phi}$, thus contradicting the validity of $\Delta \vdash A$. But if $\mathbf{\Phi} \cup \{\neg A\}$ is inconsistent, we get $\Delta, \neg A \vdash \bot$ and thus $\Delta \vdash A$ in NK_0. □

Theorem A.13 is actually **stronger** than Thm A.9. This is best seen via the usual (purely model-theoretic) **Compactness Theorem** for (classical) propositional logic. See *e.g.* [Bus98a, 1.1.5] (see also [vD04, Ex. 5.8]).

Theorem A.14 (Compactness)**.** *If $\mathbf{\Phi} \models A$ then there is a **finite** $\mathbf{\Phi}_0 \subseteq \mathbf{\Phi}$ such that $\mathbf{\Phi}_0 \models A$.*

Theorem A.14 should not be confused with the following trivial consequence of Def. A.11.(1):

- If $\mathbf{\Phi} \vdash A$, then there is a **finite** $\mathbf{\Phi}_0 \subseteq S$ such that $\mathbf{\Phi}_0 \vdash A$.

It is not difficult to see that Thm. A.13 is equivalent to (the conjunction of Thm. A.9 with) Thm. A.14.

Proof. Assume Thm. A.13. We already noted that it gives Thm. A.9, so we only discuss Thm. A.14. Consider $\mathbf{\Phi}$ and A such that $\mathbf{\Phi} \models A$. By definition, there is no valuation v such that $v \models \neg A$ and $v \models \mathbf{\Phi}$. It thus follows from Thm. A.13 that $\mathbf{\Phi} \cup \{\neg A\}$ is inconsistent, *i.e.* that $\mathbf{\Phi}, \neg A \vdash \bot$. Hence there is a finite $\mathbf{\Phi}_0 \subseteq \mathbf{\Phi}$ such that $\mathbf{\Phi}_0, \neg A \vdash \bot$, that is $\mathbf{\Phi}_0 \vdash A$ in NK_0 and thus $\mathbf{\Phi}_0 \models A$.

Conversely, assume Thm. A.14. Let $\mathbf{\Phi}$ be a consistent set of formulae. Assume toward a contradiction that $\mathbf{\Phi} \models \bot$. Hence by Thm. A.14 there would be $A_1, \dots, A_n \in \mathbf{\Phi}$ such that the sequent $A_1, \dots, A_n \vdash \bot$ is valid, so that $A_1, \dots, A_n \vdash \bot$ in NK_0 by Thm. A.9. But the latter would imply $\mathbf{\Phi} \vdash \bot$, a contradiction. It follows that $\mathbf{\Phi} \not\models \bot$, which by definition gives a valuation v such that $v \models \bot$ and $v \models \mathbf{\Phi}$. □

The Compactness Theorem A.14 is the core of Thm. A.13 in terms of infinite combinatorics. It is typically proved using the **Prime Ideal Theorem**, which itself follows from **Zorn's Lemma**, see *e.g.* [DP02, §10.16–10.19 & §11.16]. The precise statement of [vD04, Thm. 1.5.13] avoids the Prime Ideal Theorem since it restricts to **countable** languages (but still requires some infinitary combinatorics, see [Sim10, §IV.3]). See also [Bus98a, 1.1.5 & 1.1.6] for further comments.

The name "compactness" for Thm. A.14 suggests connections with topological compactness, and indeed, Thm. A.14 can be seen as a consequence of the compactness of products of compact spaces. The latter result is the famous **Tychonoff Theorem**, see *e.g.* [Run05, Thm. 3.3.21] or [Wil70, Thm. 17.8]. See *e.g.* [Run05, §3.3] or [Wil70, §8 & §17] for more on topological compactness and product spaces. In its general form, Tychonoff's Theorem is equivalent to the Axiom of Choice (see *e.g.* [Wil70, Ex. 17 O] or [Joh82, Notes on §III.1]).

Topological Proof of the Compactness Theorem A.14. It is easy to see that the Compactness Theorem A.14 follows from Tychonoff's Theorem (see also [Bus98a, 1.1.6]).

Proof. Write AP for the set of atomic propositions. In contrast with Def. 2.10 (§2.1.1) we see valuations $v \subseteq \mathrm{AP}$ as characteristic functions $\mathrm{AP} \to \mathbf{2}$ (that we still denote v).

Consider the product space $\mathbf{2}^{\mathrm{AP}}$, with $\mathbf{2}$ discrete (see *e.g.* [Run05, Def. 3.3.19] or [Wil70, Chap. 3, §8]). It follows from the Tychonoff's Theorem (see *e.g.* [Run05, Thm. 3.3.21] or [Wil70, Thm. 17.8]) that $\mathbf{2}^{\mathrm{AP}}$ is compact since $\mathbf{2}$ is compact. Note that a given formula A over AP only involves a finite number of propositional variables over AP, so that the set $\mathcal{U}_A := \{v \in \mathbf{2}^{\mathrm{AP}} \mid v \models A\}$ is a clopen subset of $\mathbf{2}^{\mathrm{AP}}$.

Assume now that $\mathbf{\Phi} \models A$, where $\mathbf{\Phi}$ is an arbitrary set of formulae (over AP). Given $v \in \mathbf{2}^{\mathrm{AP}}$, note that $v \not\models \mathbf{\Phi}$ if and only if v belongs to the open set $\bigcup\{\mathcal{U}_{\neg B} \mid B \in \mathbf{\Phi}\}$. Hence, $\mathbf{\Phi} \models A$ means that $\mathbf{2}^{\mathrm{AP}}$ is covered by the family of opens sets $\{\mathcal{U}_A\} \cup \{\mathcal{U}_{\neg B} \mid B \in \mathbf{\Phi}\}$. Since $\mathbf{2}^{\mathrm{AP}}$ is compact, it admits a finite subcover, say $\{\mathcal{U}_A\} \cup \{\mathcal{U}_{\neg B} \mid B \in \mathbf{\Phi}_0\}$ for some **finite** $\mathbf{\Phi}_0 \subseteq \mathbf{\Phi}$. But $\mathbf{2}^{\mathrm{AP}} \subseteq \{\mathcal{U}_A\} \cup \{\mathcal{U}_{\neg B} \mid B \in \mathbf{\Phi}_0\}$ precisely means that $\mathbf{\Phi}_0 \models A$. □