Proof Theory

Helmut Schwichtenberg

Mathematisches Institut der Universität München
Sommersemester 1999

Preface

These notes represent a course on proof theory held in the Sommersemester 1999 at Mathematisches Institut der Universität München. It was assumed that the audience already had a reasonable knowledge of Mathematical Logic. In particular, some subjects commonly included in an introductory proof theory course have been omitted, most notably the analysis of initial cases of transfinite induction in arithmetic. One reason was that this subject in Munich is contained in the standard course on Mathematical Logic, hence there was no need to repeat.

München, September 1999
Helmut Schwichtenberg

Table of Contents

1. Logic via Natural Deduction

The main subject of this course is mathematical proof. In this introductory chapter we deal with the basics of formalizing such proofs. The system we pick for the representation of proofs is Gentzen's natural deduction, from [15]. Our reasons for this choice are twofold. First, as the name says this is a *natural* notion of formal proof, which means that the way proofs are represented corresponds very much to the way a careful mathematician writing out all details of an argument would go anyway. Second, formal proofs in natural deduction are closely related (via the so-called Curry-Howard correspondence) to terms in typed λ-calculus. This provides us not only with a compact notation for logical derivations (which otherwise tend to become somewhat unmanagable tree-like structures), but also opens up a route to applying the computational techniques which underpin λ-calculus.

Apart from classical logic we will also deal with more constructive logics: minimal and intuitionistic logic. This will reveal some interesting aspects of proofs, e.g. that it is possible und useful to distinguish beween existential proofs that actually construct witnessing objects, and others that don't. As an example, consider the following proposition.

There are irrational numbers a, b such that a^b is rational.

This can be proved as follows, by cases.

Case if $\sqrt{2}^{\sqrt{2}}$ is rational, then choose $a = \sqrt{2}$ and $b = \sqrt{2}$. Then a, b are irrational and by assumption a^b is rational.

Case if $\sqrt{2}^{\sqrt{2}}$ is irrational, then choose $a = \sqrt{2}^{\sqrt{2}}$ and $b = \sqrt{2}$. Then by assumption a, b are irrational and

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \left(\sqrt{2}\right)^2 = 2$$

is rational. □

As long as we have not decided whether $\sqrt{2}^{\sqrt{2}}$ is rational, we do not know which numbers a, b we must take. Hence we have an example of an existence proof which does not provide an instance.

After a short reminder of first order languages we introduce Gentzen's system of natural deduction. The logical rules come in pairs: we have introduction and elimination rules for each of the logical connectives $\wedge$, $\to$ and $\forall$. This results in a system called *minimal logic*, introduced by Johannsen [20] in 1937. If we then go on and give a special status to the particular propositional symbol $\bot$ for falsity , we can embed intuitionistic logic (requiring ex-falso-quodlibet) and classical logic (requiring the principle of indirect proof, also called stability). All this is easy, since we restrict to the language based on $\{\to, \wedge, \forall\}$, and can introduce disjunction $\vee$ and the existential quantifier $\exists$ only via their (classical) de Morgan definitions. For these the usual introduction and elimination rules can then be derived. Finally we extend the language by the strong (constructive) existential quantifier $\exists^*$. Then in cases where an existential proof actually contains the construction of an example this fact can be expressed properly in the logical language.

1.1 First Order Languages

Let a countable infinite set $\{ v_i \mid i \in \mathbb{N} \}$ of *variables* be given; they will be denoted by x, y, z. A first order language $\mathcal{L}$ then is determined by its *signature*, which is to mean the following.

- For every natural number $n \geq 0$ a (possible empty) set $\mathsf{Rel}_{\mathcal{L}}^{(n)}$ of n-ary *relation symbols* (also called *predicate symbols*). 0-ary relation symbols are called *propositional symbols*. We assume that a particular propositional symbol $\bot$ (read "falsum" or "bottom") is present.
- For every natural number $n \geq 0$ a (possible empty) set $\mathsf{Fun}_{\mathcal{L}}^{(n)}$ of n-ary *function symbols*. 0-ary function symbols are called *constants*.

We assume that the set of variables and all sets $\mathsf{Rel}_{\mathcal{L}}^{(n)}$ and $\mathsf{Fun}_{\mathcal{L}}^{(n)}$ are disjoint.

For instance the language $\mathcal{L}_G$ of group theory is determined by the signature consisting of the following relation and function symbols: the group operation $\circ$ (a binary function symbol), the unit e (a constant), the inverse operation $^{-1}$ (a unary function symbol) and finally equality $=$ (a binary relation symbol).

$\mathcal{L}$-terms are inductively defined as follows.

- Every variable is an $\mathcal{L}$-term.
- Every constant of $\mathcal{L}$ is an $\mathcal{L}$-term.
- If $t_1, \dots, t_n$ are $\mathcal{L}$-terms and f is an n-ary function symbol of $\mathcal{L}$ with $n \geq 1$, then $f(t_1, \dots, t_n)$ is an $\mathcal{L}$-term.

$\mathsf{Ter}_{\mathcal{L}}$ denotes the set of $\mathcal{L}$-terms. For every $\mathcal{L}$-term t we define the set $\mathsf{vars}(t)$ of all variables occurring in t by

- $\mathsf{vars}(x) := \{x\}$ for every variable x.
- $\mathsf{vars}(c) := \emptyset$ for every constant c.
- If $t_1, \dots, t_n$ are $\mathcal{L}$-terms and f is an n-ary function symbol of $\mathcal{L}$ with $n \geq 1$, then $\mathsf{vars}(f(t_1, \dots, t_n)) := \mathsf{vars}(t_1) \cup \dots \cup \mathsf{vars}(t_n)$.

t is said to be *closed* or *ground* if $\mathsf{vars}(t) = \emptyset$.

Binary function symbols are usually written in *infix notation*, e.g. $x+y$ instead of $+(x, y)$. Parentheses are only written if necessary to make clear the structure of a term. Instead of $f(t_1, \dots, t_n)$ we often write $ft_1 \dots t_n$. Terms of the language of group theory ($\mathcal{L}_G$-terms) are for instance e, $x \circ e$, $(x^{-1} \circ y^{-1})^{-1}$.

From $\mathcal{L}$-terms one constructs easily *$\mathcal{L}$-prime formulas*, also called *atomic formulas* of $\mathcal{L}$: If $t_1, \dots, t_n$ are terms and R is an n-ary relation symbol of $\mathcal{L}$, then $R(t_1, \dots, t_n)$ is an $\mathcal{L}$-prime formula; in particular $\bot$ is an $\mathcal{L}$-prime formula.

Binary relation symbols are also usually written in *infix notation*, e.g. $x < y$ instead of $<(x, y)$. Parentheses again are only written if necessary, and $Rt_1 \dots t_n$ is short for $R(t_1, \dots, t_n)$.

$\mathcal{L}$-formulas are inductively defined from $\mathcal{L}$-prime formulas by

- Every $\mathcal{L}$-prime formula is an $\mathcal{L}$-formula.
- If A and B are $\mathcal{L}$-formulas, then so is $(A \wedge B)$ and $(A \to B)$.
- If A is an $\mathcal{L}$-formula and x is a variable, then $\forall x A$ is an $\mathcal{L}$-formula.

For every $\mathcal{L}$-formula A we define the set $\mathsf{FV}(A)$ of all its *free variables* as follows.

$$\begin{aligned}
\mathsf{FV}(R(t_1, \dots, t_n)) &:= \mathsf{vars}(t_1) \cup \dots \cup \mathsf{vars}(t_n), \\
\mathsf{FV}(A \wedge B) &:= \mathsf{FV}(A) \cup \mathsf{FV}(B), \\
\mathsf{FV}(A \to B) &:= \mathsf{FV}(A) \cup \mathsf{FV}(B), \\
\mathsf{FV}(\forall x A) &:= \mathsf{FV}(A) \setminus \{x\}.
\end{aligned}$$

In case $\mathsf{FV}(A) = \emptyset$, the formula A is said to be *closed*. We write $A[\boldsymbol{x}]$ to indicate that all the variables free in A are contained in the list $\boldsymbol{x} = x_1, x_2, \dots, x_n$.

For simplicity we have restricted ourselves to the logical symbols $\wedge$, $\to$ und $\forall$. Negation, disjunction, equivalence and the existential quantifier are then defined by

$$\begin{aligned}
\neg A &:= A \to \bot, \\
A \vee B &:= \neg(\neg A \wedge \neg B), \\
A \leftrightarrow B &:= (A \to B) \wedge (B \to A), \\
\exists x A &:= \neg \forall x \neg A.
\end{aligned}$$

We usually write $t \neq s$ for $\neg(t = s)$ and $t \not< s$ for $\neg(t < s)$. To save parentheses let's agree on the following conventions.

1. The range of $\neg$ and $\forall x$ should be as small as possible. Examples:

$$\begin{array}{lll} \neg A \wedge B & \text{means} & (\neg A) \wedge B, \quad \text{not} \quad \neg(A \wedge B). \\ \forall x A \to B & \text{means} & (\forall x A) \to B, \quad \text{not} \quad \forall x(A \to B). \end{array}$$

2. The range of $\wedge$ and $\vee$ is as small as possible, taking 1. into account. Example:

$$\neg A \wedge B \to C \quad \text{means} \quad ((\neg A) \wedge B) \to C.$$

3. Iterated uses of a logical operator are always associated to the right. Example:

$$A \to B \to C \quad \text{means} \quad A \to (B \to C).$$

4. A dot immediately after $\forall x$ makes the range of that quantifier as large as possible, given the parentheses around. Example:

$$\forall x.A \to B \quad \text{means} \quad \forall x(A \to B), \quad \text{not} \quad (\forall x A) \to B.$$

We use

r, s, t	for terms
x, y, z	for variables
c	for constants
P, Q, R	for relation symbols
f, g, h	for function symbols
A, B, C, D	for formulas

We now fix a language $\mathcal{L}$, and hence speak of terms and formulas instead of $\mathcal{L}$-terms and $\mathcal{L}$-formulas.

For simplicity we identify formulas that are the same up to renaming of bound variables. Then substitution $A[x := t]$ of a term t for a variable x in a formula A can be defined rather simply.

For terms r and t we define the result $r[x := t]$ of *substitution* of t for x in r by induction on r.

$$\begin{aligned} y[x := t] &:= \begin{cases} t & \text{if } x = y \\ y & \text{otherwise} \end{cases} \\ c[x := t] &:= c \\ f(t_1, \ldots, t_n)[x := t] &:= f(t_1[x := t], \ldots, t_n[x := t]) \end{aligned}$$

For formulas A and terms t we define the result of substituting t for x in A by induction an A.

$$\begin{aligned} R(t_1, \ldots, t_n)[x := t] &:= R(t_1[x := t], \ldots, t_n[x := t]) \\ (A \wedge B)[x := t] &:= A[x := t] \wedge B[x := t] \\ (A \to B)[x := t] &:= A[x := t] \to B[x := t] \\ (\forall y A)[x := t] &:= \forall y A[x := t], \quad \text{where wlog } y \neq x \text{ and } y \notin \mathsf{vars}(t). \end{aligned}$$

1.2 Natural Deduction

We now define the notion of a *logical deduction* of a formula A. This is done via a system of "natural deduction" introduced by Gentzen [15] in 1934. Its distinctive feature is that for every logical connective there are *introduction* as well as *elimination rules*.

Let us start with some examples for natural proofs. Assume that a first order language $\mathcal{L}$ is given. For simplicity we only consider here proofs in pure logic, i.e. without assumptions (axioms) on the functions and relations used.

$$(A \wedge B \to C) \to A \to B \to C. \tag{1.1}$$

Assume $A \wedge B \to C$. To show: $A \to B \to C$. So assume A. To show: $B \to C$. So assume B. To show: C. We have $A \wedge B$, by the last two assumptions. Hence also C, by the first assumption. □

$$(A \to B \to C) \to A \wedge B \to C. \tag{1.2}$$

Assume $A \to B \to C$. To show: $A \wedge B \to C$. So assume $A \wedge B$. To show: C. We have A, by the last assumption. Hence also $B \to C$, by the first assumption. We have B, again by the last assumption. Hence also C, by the last two assumptions. □

$$(\forall x.A \to B) \to A \to \forall x B, \quad \text{if } x \notin \mathsf{FV}(A). \tag{1.3}$$

Assume $\forall x.A \to B$. To show: $A \to \forall x B$. So assume A. To show: $\forall x B$. Let x be arbitrary; note that up to now we have not made any assumptions on x. To show: B. We have $A \to B$, by the first assumption. Hence also B, by the second assumption. □

$$(A \to \forall x B) \to \forall x.A \to B, \quad \text{if } x \notin \mathsf{FV}(A). \tag{1.4}$$

Assume $A \to \forall x B$. To show: $\forall x.A \to B$. Let x be arbitrary; note that up to now we have not made any assumptions on x. To show: $A \to B$. So assume A. To show: B. We have $\forall x B$, by the first and second assumption. Hence also B. □

A characteristic feature of these proofs is that assumptions are introduced and eliminated again. At any point in time during the proof the free or "open" assumptions are known, but from time to time as the proof progresses, free assumptions may become cancelled or "closed" because of the implies-introduction rule.

We now reserve the word *proof* for the informal level; a formal representation of a proof will be called a *derivation*.

An intuitive way to communicate derivations is to view them as labelled trees. The labels of the inner nodes are the formulas derived at those points, and the labels of the leaves are formulas or terms. The labels of the nodes immediately above a node ν are the premises of the rule application, the formula at node ν is its *conclusion*. At the root of the tree we have the conclusion of the whole derivation. In natural deduction systems one works with *assumptions* affixed to some leaves of the tree; they can be *open* or else *closed*.

Any of these assumptions carries a *marker*. As markers we use *assumption variables* $\square_0, \square_1, \dots$, denoted by $u, v, w, u_0, u_1, \dots$. The (previous) variables will now often be called *object variables* to distinguish them from assumption variables. If at a later stage (i.e. at a node below an assumption) the dependency on this assumption is removed, we record this by writing down the assumption variable. Since the same assumption can be used many times (this was the case in example 1.2), the assumption marked with u (and communicated by $u\colon A$) may appear many times. However, we insist that distinct assumption formulas must have distinct markers.

An inner node of the tree is understood as the result of passing form premises to a *conclusion*, as described by a given *rule*. The label of the node then contains in addition to the conclusion also the name of the rule. In some cases the rule binds or closes an assumption variable u (and hence removes the dependency of all assumptions $u\colon A$ marked with that u). An application of the $\forall$-introduction rule similarly binds an object variable x (and hence removes the dependency on x). In both cases the bound assumption or object variable is added to the label of the node.

We now formulate the rules of natural deduction. First we have an assumption rule, that allows an arbitrary formula A to be put down together with a marker u:

$$u\colon A \quad \text{Assumption}$$

The other rules of natural deduction split into introduction and elimination rules for the logical connectives $\wedge$, $\to$ and $\forall$. For conjunction $\wedge$ we have an introduction rule $\wedge^+$ and two elimination rules $\wedge_0^-$ und $\wedge_1^-$.

$$\dfrac{\begin{matrix}\mathcal{D}_0 & \mathcal{D}_1\\ A & B\end{matrix}}{A\wedge B}\wedge^+ \qquad \dfrac{\begin{matrix}\mathcal{D}\\ A\wedge B\end{matrix}}{A}\wedge_0^- \qquad \dfrac{\begin{matrix}\mathcal{D}\\ A\wedge B\end{matrix}}{B}\wedge_1^-$$

For implication $\to$ there is an introduction rule $\to^+ u$ and an elimination rule $\to^-$, also called *modus ponens*. The left premise $A \to B$ in $\to^-$ is called *main premise*, and the right premise A *side premise*. Note that with an application of the $\to^+ u$-rule *all* assumptions above it marked with $u\colon A$ are cancelled.

$$\dfrac{\begin{matrix}[u\colon A]\\ \mathcal{D}\\ B\end{matrix}}{A\to B}\to^+ u \qquad \dfrac{\begin{matrix}\mathcal{D}_0 & \mathcal{D}_1\\ A\to B & A\end{matrix}}{B}\to^-$$

For the universal quantifier $\forall$ there is an introduction rule $\forall^+ x$ and an elimination rule $\forall^-$, whose right premise is the term t to be substituted. The rule $\forall^+ x$ is subject to the following *(eigen-) variable condition*: The derivation $\mathcal{D}$ of the premise A should not contain any open assumption with x as a free variable.

$$\dfrac{\begin{matrix}\mathcal{D}\\ A\end{matrix}}{\forall x A}\forall^+ x \qquad \dfrac{\begin{matrix}\mathcal{D} & \\ \forall x A & t\end{matrix}}{A[x := t]}\forall^-$$

We now give derivations for the example formulas (1.1) – (1.4). Since in many cases the rule used is determined by the context, we suppress in such cases the name of the rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon A\wedge B\to C \qquad \dfrac{v\colon A \quad w\colon B}{A\wedge B}}{C}}{B\to C}\to^+ w}{A\to B\to C}\to^+ v}{(A\wedge B\to C)\to A\to B\to C}\to^+ u \tag{1.1}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon A\to B\to C \qquad \dfrac{v\colon A\wedge B}{A}}{B\to C} \qquad \dfrac{v\colon A\wedge B}{B}}{C}}{A\wedge B\to C}\to^+ v}{(A\to B\to C)\to A\wedge B\to C}\to^+ u \tag{1.2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{u\colon \forall x.A\to B \qquad x}{A\to B} \qquad v\colon A}{B}}{\forall x B}x}{A\to\forall x B}\to^+ v}{(\forall x.A\to B)\to A\to\forall x B}\to^+ u \tag{1.3}$$

note here that the variable condition is satisfied: x is not free in A (and also not free in $\forall x.A \to B$).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{u\colon A\to\forall x B \qquad v\colon A}{\forall x B} \qquad x}{B}}{A\to B}\to^+ v}{\forall x.A\to B}x}{(A\to\forall x B)\to\forall x.A\to B}\to^+ u \tag{1.4}$$

Here too the variable condition is satisfied: x is not free in $A \to \forall x B$.

Notation: We write $\vdash A$ and call A *derivable* (in *minimal logic*), if there is a derivation of A without free assumptions. A formula B is called derivable from assumptions $A_1, \dots, A_n$, if there is a derivation of B with free assumptions among $A_1, \dots, A_n$. Let Γ be a (finite or infinite) set of formulas. We write $\Gamma \vdash B$ if the formula B is derivable from finitely many assumptions $A_1, \dots, A_n \in \Gamma$.

1.3 Sequent Formulation of Natural Deduction

To find the assumptions free at the node ν of an arbitrary derivation tree, one has to search through the assumptions at the leaves above ν. Those whose marker has not been cancelled on the path from the leaf to the node are free at node ν. Although it means a lot of writing it is sometimes useful to explicitely carry along the assumptions free (or open) at a node. The set of assumptions free at a node is called its "context" $\Gamma = \{u_1 : A_1, u_2 : A_2, \dots, u_n : A_n\}$ with distinct u_i. The A_i need not be distinct; this corresponds to the fact that in a natural deduction the same assumption formula may be used with different markers. Derivations then become trees each of whose nodes is labelled with a sequent of the form $\Gamma \Rightarrow B$, where Γ is a context.

With Γ, Δ or also $\Gamma\Delta$ we denote the union $\Gamma \cup \Delta$. Use of this notation should always imply that the union is *consistent* and hence a context again.

$$u : A \Rightarrow A \quad \text{Assumption}$$

$$\frac{\Gamma \Rightarrow A \qquad \Delta \Rightarrow B}{\Gamma\Delta \Rightarrow A \wedge B}\wedge^+ \qquad \frac{\Gamma \Rightarrow A \wedge B}{\Gamma \Rightarrow A}\wedge_0^- \qquad \frac{\Gamma \Rightarrow A \wedge B}{\Gamma \Rightarrow B}\wedge_1^-$$

$$\frac{\Gamma[u : A] \Rightarrow B}{\Gamma \Rightarrow A \to B}\to^+ u \qquad \frac{\Gamma \Rightarrow A \to B \qquad \Delta \Rightarrow A}{\Gamma\Delta \Rightarrow B}\to^-$$

$$\frac{\Gamma \Rightarrow A}{\Gamma \Rightarrow \forall x A}\forall^+ x \qquad \frac{\Gamma \Rightarrow \forall x A \qquad t}{\Gamma \Rightarrow A[x := t]}\forall^-$$

The notation $\Gamma[u : A]$ in rule $\to^+ u$ means that the context may or may not contain the assumption $u : A$. The variable condition of the rule $\forall^+$ now simply is that x should not be free in any formula of Γ.

1.4 Embedding of Intuitionistic and Classical Logic

In our language based on $\{\to, \wedge, \forall\}$, we obtain *intuitionistic logic* simply by adding the following axiom (ex-falso-quodlibet) for any relation symbol R distinct from $\bot$

$$\forall \boldsymbol{x}. \bot \to R\boldsymbol{x} \tag{Efq_R}$$

Similarly one obtains *classical logic* by adding for every relation symbol R distinct from $\bot$, the principle of indirect proof expressed as the so-called "stability axiom":

$$\forall \boldsymbol{x}. \neg\neg R\boldsymbol{x} \to R\boldsymbol{x} \tag{Stab_R}$$

Note that with $\bot$ for R both formulas are trivially derivable; e.g. for stability we have $\neg\neg\bot \to \bot = ((\bot \to \bot) \to \bot) \to \bot$. The desired derivation is

$$\frac{v : (\bot \to \bot) \to \bot \qquad \dfrac{u : \bot}{\bot \to \bot}\to^+ u}{\bot}$$

Let

$$\begin{aligned} \mathsf{Efq} &:= \{\, \mathsf{Efq}_R \mid R \text{ relation symbol} \neq \bot \,\}, \\ \mathsf{Stab} &:= \{\, \mathsf{Stab}_R \mid R \text{ relation symbol} \neq \bot \,\}. \end{aligned}$$

We call the formula A *classically (intuitionistically) derivable* and write $\vdash_c A$ ($\vdash_i A$) if there is a derivation of A from stability assumptions Stab_R (respectively ex-falso-quodlibet assumptions Efq_R). Similarly we define classical (intuitionistic) derivability from Γ and write $\Gamma \vdash_c A$ ($\Gamma \vdash_i A$), i.e.

$$\begin{aligned} \Gamma \vdash_i A \ &:\Longleftrightarrow \ \Gamma \cup \mathsf{Efq} \vdash A, \\ \Gamma \vdash_c A \ &:\Longleftrightarrow \ \Gamma \cup \mathsf{Stab} \vdash A. \end{aligned}$$

Lemma 1.4.1 (Ex-falso-quodlibet). *For every formula A,*

$$\vdash_i \bot \to A.$$

Proof. By induction on A we construct for every formula A a derivation $\mathcal{D}_A$ of $\bot \to A$. *Case* $R(\boldsymbol{t})$. Use Efq_R. *Case* $A \land B$.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_A\\ \bot \to A\end{array} \qquad u\colon \bot}{A} \qquad \dfrac{\begin{array}{c}\mathcal{D}_B\\ \bot \to B\end{array} \qquad u\colon \bot}{B}}{A \land B}}{\bot \to A \land B}\to^+ u$$

Case $A \to B$.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_B\\ \bot \to B\end{array} \qquad u\colon \bot}{B}}{A \to B}}{\bot \to A \to B}\to^+ u$$

Case $\forall x A$.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\mathcal{D}_A\\ \bot \to A\end{array} \qquad u\colon \bot}{A}}{\forall x A}}{\bot \to \forall x A}\to^+ u$$

This concludes the proof. □

Lemma 1.4.2 (Stability). *For every formula A (of our language based on $\{\to, \land, \forall\}$),*

$$\vdash_c \neg\neg A \to A.$$

Proof. Induction on A. For simplicity, in the derivation to be constructed we leave out applications of $\to^+$ at the end. *Case* $R(\boldsymbol{t})$. Use Stab_R. *Case* $A \land B$. Use $\vdash (\neg\neg A \to A) \to (\neg\neg B \to B) \to \neg\neg(A \land B) \to A \land B$, which in turn follows easily from $\vdash \neg\neg(A \land B) \leftrightarrow (\neg\neg A \land \neg\neg B)$ (cf. problem 1.11.2(c)). *Case* $A \to B$. Use $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$. A derivation is

$$\dfrac{u\colon \neg\neg B \to B \qquad \dfrac{\dfrac{v\colon \neg\neg(A \to B) \qquad \dfrac{\dfrac{u_1\colon \neg B \qquad \dfrac{u_2\colon A \to B \qquad w\colon A}{B}}{\bot}}{\neg(A \to B)}\to^+ u_2}{\bot}}{\neg\neg B}\to^+ u_1}{B}$$

Case $\forall x A$. Clearly it suffices to show $\vdash (\neg\neg A \to A) \to \neg\neg\forall x A \to A$. A derivation is

$$\dfrac{u\colon \neg\neg A \to A \qquad \dfrac{\dfrac{v\colon \neg\neg\forall x A \qquad \dfrac{\dfrac{u_1\colon \neg A \qquad \dfrac{u_2\colon \forall x A \qquad x}{A}}{\bot}}{\neg\forall x A}\to^+ u_2}{\bot}}{\neg\neg A}\to^+ u_1}{A}$$

This concludes the proof. □

Lemma 1.4.3. $\Gamma \vdash A \implies \Gamma \vdash_i A$ *and* $\Gamma \vdash_i A \implies \Gamma \vdash_c A$.

Proof. It suffices to show that $\vdash_c \mathsf{Efq}_R$. This can be seen as follows. For simplicity, assume that R is unary.

$$\cfrac{\cfrac{\cfrac{\cfrac{\forall x.\neg\neg Rx \to Rx \qquad x}{\neg\neg Rx \to Rx} \qquad \cfrac{u\colon \bot}{\neg\neg Rx}\,{\to^+} v^{\neg Rx}}{Rx}}{\bot \to Rx}\,{\to^+} u}{\forall x.\bot \to Rx}\,\forall^+$$

This concludes the proof. □

Note that neither of the two implications can be reversed. Counterexamples are

$$\begin{aligned} &\nvdash \bot \to P, \quad \text{but} \quad \vdash_i \bot \to P,\\ &\nvdash_i ((P \to Q) \to P) \to P, \quad \text{but} \quad \vdash_c ((P \to Q) \to P) \to P. \end{aligned}$$

Apart from $\nvdash_i ((P \to Q) \to P) \to P$ the proofs are easy. However, to prove the non-derivability of the Peirce formula $((P \to Q) \to P) \to P$ in intuitionistic logic requires a more careful study of intuitionistic derivability. We will give a prove in chapter 1.7.

Lemma 1.4.4 (Proof by cases). *$\vdash_c (A \to B) \wedge (\neg A \to B) \to B$.*

Proof. Let C abbreviate $(A \to B) \wedge (\neg A \to B)$.

$$\cfrac{\begin{matrix}\mathcal{D}_{\mathsf{Stab}}\\ \neg\neg B \to B\end{matrix} \qquad \cfrac{\cfrac{w\colon \neg B \qquad \cfrac{\cfrac{u\colon C}{\neg A \to B} \qquad \cfrac{\cfrac{w\colon \neg B \qquad \cfrac{\cfrac{u\colon C}{A \to B} \qquad v\colon A}{B}}{\bot}}{\neg A}\,{\to^+} v}{B}}{\bot}}{\neg\neg B}\,{\to^+} w}{B}$$

where $\mathcal{D}_{\mathsf{Stab}}$ comes from the stability lemma 1.4.2. □

Call two formulas A and B *equivalent* in minimal (classical, intuitionistic) logic if $\vdash A \leftrightarrow B$ ($\vdash_c A \leftrightarrow B$, $\vdash_i A \leftrightarrow B$).

Lemma 1.4.5 (Equivalence). *Let $\vdash_{mic} \in \{\vdash, \vdash_i, \vdash_c\}$. If $\vdash_{mic} A_1 \leftrightarrow A_2$, and B_2 is obtained from B_1 by replacing one subformula A_1 of B_1 by A_2, then we also have $\vdash_{mic} B_1 \leftrightarrow B_2$.*

Proof. Induction on B_1. If all of B_1 had been replaced, the claim is obvious. Otherwise B_1 must be a composed formula.

Case $C_1 \wedge D_1$. Assume the replacement takes places in C_1. We have to show $\vdash_{mic} C_1 \wedge D_1 \leftrightarrow C_2 \wedge D_1$. $\to$:

$$\cfrac{\cfrac{\begin{matrix}\mathcal{D}\\ C_1 \to C_2\end{matrix} \qquad \cfrac{C_1 \wedge D_1}{C_1}}{C_2} \qquad \cfrac{C_1 \wedge D_1}{D_1}}{C_2 \wedge D_1}$$

where $\mathcal{D}$ is known by induction hypothesis. $\leftarrow$ is proved similarly.

Case $C_1 \to D_1$. Assume the replacement takes places in C_1. We have to show $\vdash_{mic} (C_1 \to D_1) \leftrightarrow (C_2 \to D_1)$. $\to$:

$$\cfrac{\cfrac{C_1 \to D_1 \qquad \cfrac{\begin{matrix}\mathcal{D}\\ C_2 \to C_1\end{matrix} \qquad u\colon C_2}{C_1}}{D_1}}{C_2 \to D_1}\,{\to^+} u$$

where again $\mathcal{D}$ is known by induction hypothesis. $\leftarrow$ is proved similarly. Assume now that the replacement takes places in D_1. We must show $\vdash_{mic} (C_1 \to D_1) \leftrightarrow (C_1 \to D_2)$. $\to$:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ D_1 \to D_2\end{array} \qquad \dfrac{C_1 \to D_1 \qquad u\colon C_1}{D_1}}{D_2}}{C_1 \to D_2}\,{\to^+}u$$

where again $\mathcal{D}$ is known by induction hypothesis. $\leftarrow$ is proved similarly.

Case $\forall x C_1$. We must show $\vdash_{mic} \forall x C_1 \leftrightarrow \forall x C_2$. $\to$:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ C_1 \to C_2\end{array} \qquad \dfrac{\forall x C_1 \qquad x}{C_1}}{C_2}}{\forall x C_2}$$

where again $\mathcal{D}$ is known by induction hypothesis. Observe that $\mathcal{D}$ does not contain free assumptions. $\leftarrow$ is proved similarly. □

Having defined classical and intuitionistic logic out of minimal logic by adding axioms, we next show that in fact, both logics can be embedded inside minimal logic. This may seem strange, but it is due crucially to the fact that we restrict ourselves to the language based on $\{\to, \wedge, \forall\}$.

A formula A is called *negative*, if every atomic formula $R\boldsymbol{t}$ of A different from $\bot$ occurs negated, i.e. in a context $R\boldsymbol{t} \to = falsum$.

Lemma 1.4.6. *For negative A we have $\vdash \neg\neg A \to A$.*

Proof. This follows from the stability lemma using $\vdash \neg\neg\neg R(\boldsymbol{t}) \to \neg R(\boldsymbol{t})$. □

Definition 1.4.7 (Negative translation ^{g} of Gödel-Gentzen).

$$\begin{aligned} R(\boldsymbol{t})^g &:= \neg\neg R(\boldsymbol{t}) \quad \text{for } R \neq \bot, \\ \bot^g &:= \bot, \\ (A \wedge B)^g &:= A^g \wedge B^g, \\ (A \to B)^g &:= A^g \to B^g, \\ (\forall x A)^g &:= \forall x A^g. \end{aligned}$$

Theorem 1.4.8. *For all fomulas A we have*

1. $\vdash_c A \leftrightarrow A^g$,
2. $\Gamma \vdash_c A$ *iff* $\Gamma^g \vdash A^g$, *where* $\Gamma^g := \{\, B^g \mid B \in \Gamma \,\}$.

Proof. The first part follows easily from the equivalence lemma 1.4.5. For the second part, the direction from right to left is obvious. For the other direction, we argue by induction on the classical derivation. For a stability assumption $\neg\neg R(\boldsymbol{t}) \to R(\boldsymbol{t})$ we have $(\neg\neg R(\boldsymbol{t}) \to R(\boldsymbol{t}))^g = \neg\neg\neg\neg R(\boldsymbol{t}) \to \neg\neg R(\boldsymbol{t})$, and this is easily derivable. *Case* $\to^+$. Assume

$$\dfrac{\begin{array}{c}[u\colon A]\\ \mathcal{D}\\ B\end{array}}{A \to B}\,{\to^+}u$$

Then we have by induction hypothesis

$$\begin{array}{c}u\colon A^g\\ \mathcal{D}^g\\ B^g\end{array} \quad \text{hence} \quad \dfrac{\begin{array}{c}[u\colon A^g]\\ \mathcal{D}^g\\ B^g\end{array}}{A^g \to B^g}\,{\to^+}u$$

Case $\to^-$. Assume

$$\dfrac{\begin{array}{c}\mathcal{D}_0\\ A \to B\end{array} \qquad \begin{array}{c}\mathcal{D}_1\\ A\end{array}}{B}$$

Then we have by induction hypothesis

$$\begin{array}{cc} \mathcal{D}_0^g & \mathcal{D}_1^g \\ A^g \to B^g & A^g \end{array} \quad \text{hence} \quad \dfrac{\begin{array}{cc} \mathcal{D}_0^g & \mathcal{D}_1^g \\ A^g \to B^g & A^g \end{array}}{B^g}$$

The other cases are treated similarly. □

Corollary 1.4.9 (Embedding of classical logic). *For negative A,*

$$\vdash_c A \iff \vdash A.$$

Proof. By the theorem we have $\vdash_c A$ iff $\vdash A^g$. Since A is negative, every atom distinct from $\bot$ in A must occur negated, and hence in A^g it must appear in threefold negated form (as $\neg\neg\neg R(\boldsymbol{t})$). The claim follows from $\vdash \neg\neg\neg R(\boldsymbol{t}) \leftrightarrow \neg R(\boldsymbol{t})$. □

Since every formula is classically equivalent to a negative formula, we have achieved an embedding of classical logic into minimal logic.

Note that $\nvdash \neg\neg P \to P$ (as we shall show in chapter 1.7). The corollary therefore does not hold for all formulas A.

Proposition 1.4.10 (Kuroda [22]). *Let A be a formula every universal quantifier of which has its scope negated. Then we have $\vdash_c \neg A$ iff $\vdash_i \neg A$.*

Proof. $\Longleftarrow$. Obvious. $\Longrightarrow$. Assume $\vdash_c \neg A$, hence also $\vdash_c \neg A'$, where A' is obtained from A by doubly negating every atomic or universal subformula. By the corollary we can conclude $\vdash \neg A'$. Because of

$$\begin{array}{ll} \vdash \neg\neg\bot \leftrightarrow \bot & \text{(cf. the proof of the stability lemma)} \\ \vdash \neg\neg(B \wedge C) \leftrightarrow (\neg\neg B \wedge \neg\neg C) & \text{(cf. problem 1.11.2(c))} \\ \vdash_i \neg\neg(B \to C) \leftrightarrow (\neg\neg B \to \neg\neg C) & \text{(cf. problems 1.11.2(d) and 1.11.4)} \end{array}$$

we can pull double negations outside, over $\wedge$ and $\to$. Since the kernels of universal formulas and also the whole formula are negated and generally $\vdash \neg\neg\neg B \leftrightarrow \neg B$, it follows that $\vdash_i \neg A$. □

Note that the proposition does not hold for minimal logic; a counterexample is $\nvdash \neg\neg(\neg\neg P \to P)$.

1.5 Derived Rules for Disjunction and Existence

Disjunction and existence have been defined by

$$A \vee B := \neg(\neg A \wedge \neg B),$$
$$\exists x A := \neg\forall x \neg A.$$

However, derivations of formulas containing $\vee, \exists$ can often be found more easily if one does not expand these definitions but rather works directly with "*derived rules*" for $\vee, \exists$. By this we mean the following rules.

For disjunction $\vee$ there are two introduction rules $\vee_0^+, \vee_1^+$ and one elimination rule $\vee^- uv$.

$$\dfrac{\begin{array}{c}\mathcal{D}\\ A\end{array}}{A \vee B}\,\vee_1^+ \qquad \dfrac{\begin{array}{c}\mathcal{D}\\ B\end{array}}{A \vee B}\,\vee_0^+ \qquad \dfrac{\begin{array}{ccc} & [u\colon A] & [v\colon B] \\ \mathcal{D} & \mathcal{D}_0 & \mathcal{D}_1 \\ A \vee B & C & C \end{array}}{C}\,\vee^- uv$$

For the existential quantifier $\exists$ there is an introduction rule $\exists^+$ whose right premise is the witnessing term t, and an elimination rule $\exists^- xu$ subject to the following *(eigen-) variable condition*: The derivation $\mathcal{D}_0$ should not contain any open assumptions apart from $u\colon A$ whose assumption formula contains x free, and moreover B should not contain the variable x free,

$$\cfrac{\begin{array}{c}\mathcal{D}\\ A[x:=t]\end{array} \qquad t}{\exists x A}\,\exists^+ \qquad\qquad \cfrac{\begin{array}{c}\mathcal{D}\\ \exists x A\end{array} \qquad \begin{array}{c}[u\colon A]\\ \mathcal{D}_0\\ B\end{array}}{B}\,\exists^- x u$$

The double line is to mean here that this is not a primitive rule of our logic, but rather that at this place a piece of derivation should be inserted in such a way that a correct derivation results. In what follows we spell out for each of the derived rules the piece of derivation to be substituted.

$\vee_0^+$ ($\vee_1^+$ is treated similarly):

$$\dfrac{\dfrac{\dfrac{u\colon \neg A\wedge\neg B}{\neg A} \qquad \begin{array}{c}\mathcal{D}\\ A\end{array}}{\bot}}{\neg(\neg A\wedge \neg B)}\to^+ u$$

$\vee^- uv$:

$$\dfrac{\begin{array}{c}\mathcal{D}_{\mathsf{Stab}}\\ \neg\neg C\to C\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \neg(\neg A\wedge\neg B)\end{array} \qquad \dfrac{\dfrac{\dfrac{w\colon\neg C \qquad \begin{array}{c}[u\colon A]\\ \mathcal{D}_0\\ C\end{array}}{\bot}}{\neg A}\to^+u \qquad \dfrac{\dfrac{w\colon\neg C \qquad \begin{array}{c}[v\colon B]\\ \mathcal{D}_1\\ C\end{array}}{\bot}}{\neg B}\to^+v}{\neg A\wedge\neg B}}{\bot}}{\neg\neg C}\to^+w}{C}$$

$\exists^+$:

$$\dfrac{\dfrac{\dfrac{u\colon\forall x\neg A \qquad t}{\neg A[x:=t]} \qquad \begin{array}{c}\mathcal{D}\\ A[x:=t]\end{array}}{\bot}}{\neg\forall x\neg A}\to^+u$$

$\exists^- xu$:

$$\dfrac{\begin{array}{c}\mathcal{D}_{\mathsf{Stab}}\\ \neg\neg B\to B\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}\mathcal{D}\\ \neg\forall x\neg A\end{array} \qquad \dfrac{\dfrac{\dfrac{v\colon\neg B \qquad \begin{array}{c}[u\colon A]\\ \mathcal{D}_0\\ B\end{array}}{\bot}}{\neg A}\to^+u}{\forall x\neg A}}{\bot}}{\neg\neg B}\to^+v}{B}$$

Alternatively, one can also achieve the comfort of the derived rules for $\vee, \exists$ by using the following (derivable) schemes as axioms.

$$A \to A \vee B, \quad B \to A \vee B.$$
$$A \vee B \to (A \to C) \to (B \to C) \to C.$$
$$A \to \exists x A.$$
$$\exists x A \to (\forall x. A \to B) \to B, \quad \text{if } x \notin \mathsf{FV}(B).$$

1.6 Strong Disjunction and Existence

If one wants to use an existence proof for purposes of program extraction, then it is not advisable only to allow the *weak* or *classical* existial quantifier $\exists$ (defined by $\neg\forall\neg$) treated up to now. Rather, we need a *strong* or *constructive* existial quantifier $\exists^*$. Then in cases where an existential proof actually contains the construction of an example, this fact can be expressed properly in the logical language.

Similarly one can introduce the *strong* or *constructive* disjunction $\vee^*$. However, in applications we will normally have a ground type ι available and could then define

$$A \vee^* B \quad := \quad \exists^* n.(n = 0 \to A) \wedge (n \neq 0 \to B).$$

Let us review what effect the addition of $\vee^*, \exists^*$ has to our previous development. We extend our notion of an $\mathcal{L}$-formula by adding the clauses $A \vee^* B$ and $\exists^* xA$. The definition of the set of free variables and of substitution are extended in the obvious way, also our conventions concerning parentheses. The logic of $\vee^*$ and $\exists^*$ could be axiomatized by rules for similar to the derived rules for the corresponding classical connectives given above. However, we prefer not to extend the concept of natural deduction and rather axiomatize $\vee^*$ and $\exists^*$ by the universal closures of the following formulas.

$$\begin{aligned}
&A \to A \vee^* B, \quad B \to A \vee^* B.\\
&A \vee^* B \to (A \to C) \to (B \to C) \to C.\\
&A \to \exists^* xA.\\
&\exists^* xA \to (\forall x.A \to B) \to B, \quad \text{if } x \notin \mathsf{FV}(B).
\end{aligned}$$

These axiom schemes are denoted by $\vee_0^{*+}$, $\vee_1^{*+}$, $\vee^{*-}$, $\exists^{*+}$ und $\exists^{*-}$. For A in the language extended with $\vee^*$ and $\exists^*$, we write $\vdash A$ and call A *derivable* (in *minimal logic*), if there is a derivation of A form instances of these axiom schemes.

Embedding of intuitionistic logic can be done as before, simply by adding the ex-falso-quodlibet-axioms Efq_R, i.e. $\forall \boldsymbol{x}.\bot \to R\boldsymbol{x}$. The ex-falso-quodlibet lemma 1.4.1 can easily be extended:

Lemma 1.6.1 (Ex-falso-quodlibet). *For every formula A in the language based on $\{\to, \wedge, \vee^*, \forall, \exists^*\}$,*

$$\vdash_i \bot \to A.$$

Proof. We only have to add cases for $A \vee^* B$ and $\exists^* xA$, which are obvious. □

1.7 Completeness for Classical, Minimal and Intuitionistic Logic

It is an obvious question to ask whether our logical rules suffice, i.e. whether we have forgotten any important rules. To answer this question we have to fix the *meaning* of a formula, i.e. we have to give a semantics. In this section we first give a standard completeness proof for classical logic with respect to ordinary semantics. Then we present a notion of a structure (due to Beth and Kripke) appropriate for minimal and intuitionistic logic, and prove completeness for both. This will enable us to e.g. prove the non-derivability of the Peirce formula $((P \to Q) \to P) \to P$ in intuitionistic logic.

Completeness for Classical Logic

We begin with a proof of completeness for classical logic, using a standard argument (due to Beth, Hintikka and Schütte), i.e. by searching simultaneously for a derivation and a countermodel. Throughout we restrict attention to *countable* languages $\mathcal{L}$, i.e. we assume that in $\mathcal{L}$ there are at most countably many relation and function symbols.

First we have to fix the notions of the meaning of a term and a formula. To this end let us define the notion of an $\mathcal{L}$-structure. $\mathcal{M} = (D, I)$ is an $\mathcal{L}$-*prestructure* if D is a nonempty set (the *carrier set* of $\mathcal{M}$) and I is a mapping (*interpretation*), assigning to every n-ary function symbol f of $\mathcal{L}$ a function $I(f)\colon D^n \to D$. If $n = 0$, then $I(f)$ is an element of D. $\mathcal{M} = (D, I_0, I_1)$ is an $\mathcal{L}$-*structure* if (D, I_0) is an $\mathcal{L}$-prestructure and I_1 is a mapping assigning to every n-ary relation symbol R of $\mathcal{L}$ an n-ary relation $I_1(R) \subseteq D^n$. If $n = 0$, then $I_1(R)$ is either true or false; in particular, $I_1(\bot)$ is false.

If $\mathcal{M} = (D, I)$ or (D, I_0, I_1), then we often write $|\mathcal{M}|$ for the carrier set D of $\mathcal{M}$, and $f^{\mathcal{M}}$ or $R^{\mathcal{M}}$ for the interpretations $I_0(f)$ and $I_1(R)$ of the function- or relation symbols.

An *environment* η in the nonempty set D assigns to every variable $x \in \mathsf{dom}(\eta)$ a value $\eta(x) \in D$. Finite environments can be written as $[a_1/x_1, \ldots, a_n/x_n]$ with distinct $x_1, \ldots, x_n$. Let η_x^a denote the environment whose value at x is set to a.

Now let an $\mathcal{L}$-prestructure $\mathcal{M}$ and an environment η in $|\mathcal{M}|$ be given. We define a homomorphic continuation of η (also denoted by η) to the set $\mathsf{Ter}_{\mathcal{L}}$ of $\mathcal{L}$-terms t with $\mathsf{vars}(t) \subseteq \mathsf{dom}(\eta)$ by

$$\eta(c) := c^{\mathcal{M}},$$
$$\eta(f(t_1, \dots, t_n)) := f^{\mathcal{M}}(\eta(t_1), \dots, \eta(t_n)).$$

Note that the extension of η depends on $\mathcal{M}$; we therefore write $t^{\mathcal{M}}[\eta]$ for $\eta(t)$.

For every $\mathcal{L}$-structure $\mathcal{M}$, environment η in $|\mathcal{M}|$ and $\mathcal{L}$-formula A with $\mathsf{FV}(A) \subseteq \mathsf{dom}(\eta)$ we define $\mathcal{M} \models A[\eta]$ (A is *valid* in $\mathcal{M}$ w.r.t. environment η) by induction on A.

$$\begin{aligned}
&\mathcal{M} \models R(t_1, \dots, t_n)[\eta] \;:\Longleftrightarrow\; (\eta(t_1), \dots, \eta(t_n)) \in R^{\mathcal{M}} \quad (R \neq \bot).\\
&\mathcal{M} \not\models \bot[\eta].\\
&\mathcal{M} \models (A \wedge B)[\eta] \;:\Longleftrightarrow\; \mathcal{M} \models A[\eta] \text{ and } \mathcal{M} \models B[\eta].\\
&\mathcal{M} \models (A \to B)[\eta] \;:\Longleftrightarrow\; \text{if } \mathcal{M} \models A[\eta], \text{ then } \mathcal{M} \models B[\eta].\\
&\mathcal{M} \models (\forall x A)[\eta] \;:\Longleftrightarrow\; \text{for all } a \in |\mathcal{M}| \text{ we have } \mathcal{M} \models A[\eta_x^a].
\end{aligned}$$

For a set Γ of $\mathcal{L}$-formulas we write $\mathcal{M} \models \Gamma[\eta]$ if $\mathcal{M} \models A[\eta]$ for all $A \in \Gamma$.

Lemma 1.7.1 (Coincidence). *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, t an $\mathcal{L}$-term, A an $\mathcal{L}$-formula and η, ξ environments in $|\mathcal{M}|$.*

1. *If $\eta(x) = \xi(x)$ for all $x \in \mathsf{vars}(t)$, then $\eta(t) = \xi(t)$.*
2. *If $\eta(x) = \xi(x)$ for all $x \in \mathsf{FV}(A)$, then $\mathcal{M} \models A[\eta]$ if and only if $\mathcal{M} \models A[\xi]$.*

Proof. Induction on terms and formulas. □

Lemma 1.7.2 (Substitution). *Let $\mathcal{M}$ be an $\mathcal{L}$-structure, t, r $\mathcal{L}$-terms, A an $\mathcal{L}$-formula and η an environment in $|\mathcal{M}|$. Then*

1. *$\eta(r[x := t]) = \eta_x^{\eta(t)}(r)$.*
2. *$\mathcal{M} \models A[x := t][\eta] \iff \mathcal{M} \models A[\eta_x^{\eta(t)}]$.*

Proof. Induction on terms and formulas. □

For an arbitrary (wlog closed) $\mathcal{L}$-formula A we shall show that A is derivable in classical logic if and only if A is valid in all $\mathcal{L}$ structures.

To simplify the argument we first prove that we can restrict attention to the language $\{\to, \forall\}$.

Lemma 1.7.3 (Elimination of $\wedge$). *For every formula A in the language based on $\{\to, \wedge, \forall\}$ we can find formulas $A_1, \dots, A_n$ without $\wedge$ such that $\vdash A \leftrightarrow \bigwedge\!\!\!\bigwedge_{i=1}^n A_i$.*

Proof. Induction on A. *Case* $R\vec{t}$. Choose $n = 1$ and $A_1 := R\vec{t}$. *Case* $A \wedge B$. By induction hypothesis we have $A_1, \dots, A_n$ and $B_1, \dots, B_m$. Choose $A_1, \dots, A_n, B_1, \dots, B_m$. *Case* $A \to B$. By induction hypothesis again we have $A_1, \dots, A_n$ and $B_1, \dots, B_m$. To simplify notation assume $n = 2$ and $m = 3$. Then we have

$$\begin{aligned}
&\vdash (A_1 \wedge A_2 \to B_1 \wedge B_2 \wedge B_3)\\
&\leftrightarrow (A_1 \to A_2 \to B_1) \wedge (A_1 \to A_2 \to B_2) \wedge (A_1 \to A_2 \to B_3).
\end{aligned}$$

Case $\forall x A$. By induction hypothesis for A we have $A_1, \dots, A_n$. Choose $\forall x A_1, \dots, \forall x A_n$, for

$$\vdash \forall x \bigwedge\!\!\!\bigwedge_{i=1}^{n} A_i \leftrightarrow \bigwedge\!\!\!\bigwedge_{i=1}^{n} \forall x A_i.$$ □

For the proof of the completeness theorem we will make use of *König's lemma*. Let S be a nonempty set. A *node* over S is a finite sequence $k = \langle a_0, a_1, \dots, a_{n-1} \rangle$ of elements $a_i \in S$; n is the *length* $\mathsf{lh}(k)$ of k. We write $k \preceq k'$ if k is an initial segment of k'. A *tree* over S is a set of nodes over S closed under the formation of initial segments. A tree T is *finitely branching*, if every $k \in T$ has only finitely many immediate successors in T. A tree T is *unbounded*, if for every $n \in \mathbb{N}$ there is a node $k \in T$ such that $\mathsf{lh}(k) = n$. A *branch* in a tree T is a linearly ordered (by $\preceq$) subtree of T. Every infinite branch is determined by a sequence $(a_n)_{n \in \mathbb{N}}$ of elements of S, whose finite initial segments form the nodes of the branch.

Lemma 1.7.4 (König). *Every finitely branching unbounded tree has an infinite branch.*

Proof. Let T be a finitely branching unbounded tree over a set S. Define recursively elements $a_n \in S$ as follows. a_n is some $a \in S$ such that $\langle a_0, a_1, \dots, a_{n-1}, a\rangle$ has arbitrarily long continuations in T. □

Note that without the assumption that T is finitely branching König's lemma is false. A counterexample is the set T of all nodes $\langle n, 0, \dots, 0\rangle$ with n occurrences of 0. Clearly T is unbounded, but has no infinite branch.

Let $\Gamma \cup \{A\}$ be a set of $\mathcal{L}$-formulas. We write $\Gamma \models A$ (*Γ semantically implies A*), if for all $\mathcal{L}$-structures $\mathcal{M}$ and all environments η in $|\mathcal{M}|$, $\mathcal{M} \models B[\eta]$ for all $B \in \Gamma$ implies $\mathcal{M} \models A[\eta]$.

Theorem 1.7.5 (Correctness).

$$\Gamma \vdash_c A \implies \Gamma \models A.$$

Proof. Induction on derivations. □

We now prove the converse.

Theorem 1.7.6 (Completeness). *Let $\Gamma \cup \{A\}$ be a set of $\mathcal{L}$-formulas. Then*

$$\Gamma \models A \implies \Gamma \vdash_c A.$$

Proof. We make use of an approach due to Beth, Hintikka and Schütte, that is, we construct a universal search tree (i.e., independent of Γ and A), giving rise to either a (classical) derivation of A from Γ or a counterexample, i.e. a model of $\Gamma \cup \{\neg A\}$.

To define the universal search tree we need at certain points new variables. Clearly we can assume that there is a countably infinite set V of variables not in $\Gamma \cup \{A\}$. Let AllFor be the set of universal formulas and Ter the set of all terms of $\mathcal{L}$. Since $\mathcal{L}$ is countable, we can assume that we have a fixed enumeration of $\mathsf{AllFor} \times \mathsf{Ter}$; write $(\forall y C, s) \prec (\forall x B, t)$ if $(\forall y C, s)$ precedes $(\forall x B, t)$ in this enumeration. We now can define recursively an injective mapping $\mathsf{new}\colon \mathsf{AllFor} \times \mathsf{Ter} \to V$ such that $\mathsf{new}(\forall x B, t)$ does not occur in any $(\forall y C, s)$ with $(\forall y C, s) \preceq (\forall x B, t)$.

Recall that by lemma 1.7.3 we can assume that neither Γ nor A contains the connective $\wedge$. Let $\mathsf{AtomFor}$ be the set of atomic formulas and ImpFor be the set of implications of $\mathcal{L}$. Consider

$$I := \mathsf{AtomFor} \cup \mathsf{ImpFor} \cup (\mathsf{AllFor} \times \mathsf{Ter})$$

as an index set. For every $F \in I$ and $k \in \{0,1,2\}$ define a finite set $\Delta_k(F)$ of formulas by

$$\Delta_k(B) := \begin{cases} \{\neg B\} & \text{if } k = 0 \\ \{B\} & \text{if } k \in \{1,2\} \end{cases} \qquad \text{for } B \text{ atomic}$$

$$\Delta_k(B \to C) := \begin{cases} \{\neg(B \to C), B, \neg C\} & \text{if } k = 0 \\ \{B \to C, \neg B\} & \text{if } k = 1 \\ \{B \to C, C\} & \text{if } k = 2 \end{cases}$$

$$\Delta_k(\forall x B, t) := \begin{cases} \{\neg \forall x B, \neg B[x := y]\} & \text{if } k = 0 \text{ and } y = \mathsf{new}(\forall x B, t) \\ \{\forall x B, B[x := t]\} & \text{if } k \in \{1,2\}. \end{cases}$$

A partial mapping α from I to $\{0,1,2\}$ is called *search path* or just *path*. For every search path α we define a set Γ_α of formulas by

$$\Gamma_\alpha := \bigcup_{F \in \mathsf{dom}(\alpha)} \Delta_{\alpha(F)}(F).$$

α is called *blocked* (w.r.t. our given Γ and A) if

- $A \in \Gamma_\alpha$, or $\neg B \in \Gamma_\alpha$ for some $B \in \Gamma$, or
- $B, \neg B \in \Gamma_\alpha$ for some B, or
- $\bot \in \Gamma_\alpha$.

Let α_F^k or $\alpha[k/F]$ denote the path whose value at F is set to k.

Positive main case. There is an $I_0 \subseteq_{\mathsf{fin}} I$ such that all search paths α through I_0 (i.e. with $\mathsf{dom}(\alpha) = I_0$) are blocked.

We show that in this case $\Gamma \vdash_c A$. To see this order the finite set I_0 such that we first have all elements from $\mathsf{AllFor} \times \mathsf{Ter}$ ordered by $\prec$ and afterwards all formulas from $\mathsf{AtomFor}$ and ImpFor. Let J range over subsets of I_0 that form an initial segment w.r.t. this ordering. We show that for every such J and all α through J we have $\Gamma_\alpha \cup \Gamma \vdash_c A$, by induction on the number of elements in $I_0 \setminus J$. The claim $\Gamma \vdash_c A$ then follows for $J = \emptyset$.

Base $J = I_0$. By assumption α is blocked; hence $\Gamma_\alpha \cup \Gamma \vdash_c A$. *Step.* We distinguish three cases.

Case $\{B\} \dot\cup J$ with B atomic. Let α be a search path through J. By induction hypothesis we have $\Gamma_{\alpha[k/B]} \cup \Gamma \vdash_c A$ for every $k \in \{0,1,2\}$, so

$$\{\neg B\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A \quad \text{and} \quad \{B\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A,$$

and hence

$$\Gamma_\alpha \cup \Gamma \vdash_c (B \to A) \wedge (\neg B \to A).$$

With $\vdash_c (B \to A) \wedge (\neg B \to A) \to A$ (by the cases lemma 1.4.4) we obtain $\Gamma_\alpha \cup \Gamma \vdash_c A$.

Case $\{B \to C\} \dot\cup J$. Let α be a search path through J. By induction hypothesis we have $\Gamma_{\alpha[k/B\to C]} \cup \Gamma \vdash_c A$ for every $k \in \{0,1,2\}$, so

$$\{\neg(B \to C), B, \neg C\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A, \tag{1.5}$$

$$\{B \to C, \neg B\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A, \tag{1.6}$$

$$\{B \to C, C\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A. \tag{1.7}$$

Because of $\vdash_c \neg(B \to C) \to B \wedge \neg C$ (cf. problem 1.11.5) from (1.5) we obtain

$$\{\neg(B \to C)\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A. \tag{1.8}$$

Because of $\vdash_c (\neg B \to A) \wedge (C \to A) \to (B \to C) \to A$ (cf. problem 1.11.5) from (1.6) and (1.7) we get

$$\{B \to C\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A. \tag{1.9}$$

From (1.8) and (1.9) we obtain $\Gamma_\alpha \cup \Gamma \vdash_c A$, using the cases lemma 1.4.4 again.

Case $\{(\forall x B, t)\} \dot\cup J$ with $J \subseteq \mathsf{AllFor} \times \mathsf{Ter}$ and $(\forall x B, t) \succ (\forall y C, s)$ for all $(\forall y C, s) \in J$. Let α be a search path through J. By induction hypothesis $\Gamma_{\alpha[k/(\forall x B,t)]} \cup \Gamma \vdash_c A$ for every $k \in \{0,1,2\}$, so

$$\{\neg \forall x B, \neg B[x := y]\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A,$$

$$\{\forall x B, B[x := t]\} \cup \Gamma_\alpha \cup \Gamma \vdash_c A,$$

where y is not free in $\forall x B$, Γ_α, Γ and A. Using

$$\vdash_c (\forall x B \to A) \to \forall y(\neg B[x := y] \to A) \to A$$

for y not free in $\forall x B$, A (cf. problem 1.11.5) we obtain $\Gamma_\alpha \cup \Gamma \vdash_c A$.

Negative main case. For every $I_0 \subseteq_{\mathsf{fin}} I$ there is a non-blocked search path α through I_0.

We show that there is a non-blocked search path α through all of I. From α we will construct a counterexample to the assumption of the theorem.

Since $\mathcal{L}$ is countable, so is I. Let $I = \{ F_n \mid n \in \mathbb{N} \}$ with $F_n \neq F_m$ for $n \neq m$, and

$$T := \{ k \mid \exists \alpha . \alpha \text{ non-blocked } \wedge \forall i{<}\mathsf{lh}(k). F_i \in \mathsf{dom}(\alpha) \wedge k_i = \alpha(F_i) \}.$$

Clearly T is a tree, i.e. closed under formation of initial segments. Moreover T is finitely branching, since we always have $\alpha(F) \in \{0,1,2\}$. Finally T is unbounded because of our assumption in the negative main case. By König's lemma we therefore have an infinite branch $(k_n)_{n\in\mathbb{N}}$ in T. Define $\alpha(F_n) := k_n$. Clearly α is a non-blocked search path through all of I. From

$$\Gamma_\alpha = \bigcup_{F \in I} \Delta_{\alpha(F)}(F)$$

we now construct a counterexample to the assumption of the completeness theorem. First let us note some obvious properties of Γ_α.

(a) For every formula B either $B \in \Gamma_\alpha$ or $\neg B \in \Gamma_\alpha$.
(b) $\bot \notin \Gamma_\alpha$.
(c) $A \notin \Gamma_\alpha$, and $\Gamma \subseteq \Gamma_\alpha$.
(d) If $\neg(B \to C) \in \Gamma_\alpha$, then $B \in \Gamma_\alpha$ and $\neg C \in \Gamma_\alpha$.
(e) If $B \to C \in \Gamma_\alpha$, then $\neg B \in \Gamma_\alpha$ or $C \in \Gamma_\alpha$.
(f) If $\neg\forall x B \in \Gamma_\alpha$, then $\neg B[x := y] \in \Gamma_\alpha$ for some y.
(g) If $\forall x B \in \Gamma_\alpha$, then $B[x := t] \in \Gamma_\alpha$ for all terms t.

Define an $\mathcal{L}$-structure $\mathcal{M}$ by $\mathcal{M} := (\mathsf{Ter}_\mathcal{L}, I_0, I_1)$, where I_0 is canonical (i.e. $I_0(f)(\boldsymbol{t}) := f\boldsymbol{t}$) and I_1 refers to Γ_α:

$$I_1(R) := \{\, \boldsymbol{t} \mid R\boldsymbol{t} \in \Gamma_\alpha \,\}.$$

We will show

$$\mathcal{M} \models B[\mathsf{id}] \iff B \in \Gamma_\alpha, \tag{1.10}$$

by induction on the number of logical symbols $\to, \forall$ in B. Observe that with our canonical definition of I_0 we have $t^\mathcal{M}[\mathsf{id}] = t$ for all $\mathcal{L}$-terms t. Write $\mathcal{M} \models B$ for $\mathcal{M} \models B[\mathsf{id}]$.

Case $R(\boldsymbol{t})$ $(R \neq \bot)$. $\mathcal{M} \models R(\boldsymbol{t})$ means $R(\boldsymbol{t}) \in \Gamma_\alpha$, by definition of $\mathcal{M}$.

Case $\bot$. By (b) $\bot \notin \Gamma_\alpha$.

Case $B \to C$. $\Longrightarrow$. Assume $\mathcal{M} \models B \to C$. We must show $B \to C \in \Gamma_\alpha$. By (a) it suffices to show $\neg(B \to C) \notin \Gamma_\alpha$. So assume $\neg(B \to C) \in \Gamma_\alpha$. By (d) we then have $B \in \Gamma_\alpha$ and $\neg C \in \Gamma_\alpha$, hence by (a) $C \notin \Gamma_\alpha$. By induction hypothesis we obtain $\mathcal{M} \models B$ and $\mathcal{M} \not\models C$, contradicting our assumption $\mathcal{M} \models B \to C$.

$\Longleftarrow$. Assume $B \to C \in \Gamma_\alpha$. We must show $\mathcal{M} \models B \to C$. So let $\mathcal{M} \models B$. We must show $\mathcal{M} \models C$. By induction hypothesis we have $B \in \Gamma_\alpha$, hence by (e) and (a) $C \in \Gamma_\alpha$, so $\mathcal{M} \models C$ by induction hypothesis.

Case $\forall x B$. $\Longrightarrow$. Assume $\mathcal{M} \models \forall x B$. We must show $\forall x B \in \Gamma_\alpha$. By (a) it suffices to show $\neg\forall x B \notin \Gamma_\alpha$. So assume $\neg\forall x B \in \Gamma_\alpha$. By (f) $\neg B[x := y] \in \Gamma_\alpha$ for some y, hence $\mathcal{M} \not\models B[x := y][\mathsf{id}]$ by (a) and the induction hypothesis. The substitution lemma yields $\mathcal{M} \not\models B[\mathsf{id}_x^y]$, contradicting our assumption $\mathcal{M} \models \forall x B[\mathsf{id}]$.

$\Longleftarrow$. Let $\forall x B \in \Gamma_\alpha$. We must show $\mathcal{M} \models \forall x B[\mathsf{id}]$. Pick an arbitrary $t \in \mathsf{Ter}_\mathcal{L}$. We must show $\mathcal{M} \models B[\mathsf{id}_x^t]$, i.e. $\mathcal{M} \models B[x := t][\mathsf{id}]$. From (g) we obtain $B[x := t] \in \Gamma_\alpha$, hence $\mathcal{M} \models B[x := t][\mathsf{id}]$ by induction hypothesis for $B[x := t]$.

Now we can finish the proof of the completeness theorem. Because of (c) we have $A \notin \Gamma_\alpha$ and $\Gamma \subseteq \Gamma_\alpha$, hence $\mathcal{M} \not\models A[\mathsf{id}]$ and $\mathcal{M} \models B[\mathsf{id}]$ for all $B \in \Gamma$, i.e. $\mathcal{M}$ and id yield a counterexample to the assumption $\Gamma \models A$ of the completeness theorem. □

At first sight, it seems that the completeness theorem refers to the universe of "all sets", in the definition of $\mathcal{M} \models A$. However, an inspection of the proof reveals that it suffices to refer to particular countable sets. The domain of the model is the countable set $\mathsf{Ter}_\mathcal{L}$ (which via some coding may be replaced by $\mathbb{N}$), and both the set T used in the negative main case and its infinite branch yielded by König's lemma are certainly arithmetically definable.

The completeness theorem has many important corollaries; we only note the following. Call a set Γ of $\mathcal{L}$-formulas *consistent* if $\Gamma \nvdash_c \bot$, and *satisfiable* if there is an $\mathcal{L}$ structure $\mathcal{M}$ and an environment η in $|\mathcal{M}|$ such that $\mathcal{M} \models B[\eta]$ for all $B \in \Gamma$.

Corollary 1.7.7. *Let Γ be a set of $\mathcal{L}$-formulas.*

1. *If Γ is consistent, then Γ is satisfiable.*
2. *(Compactness theorem). If every finite subset of Γ is satisfiable, then so is Γ.*

Proof. 1. From $\Gamma \nvdash_c \bot$ we obtain $\Gamma \not\models \bot$ by the completeness theorem, and this implies the satisfiability of Γ.

2. Otherwise we have $\Gamma \models \bot$, hence $\Gamma \vdash_c \bot$ by completeness, hence also $\Gamma_0 \vdash_c \bot$ for a finite subset $\Gamma_0 \subseteq \Gamma$, hence $\Gamma_0 \models \bot$ contradicting our assumption that Γ_0 has a model. □

Corollary 1.7.8 (Löwenheim, Skolem). *Let Γ be a set of $\mathcal{L}$-formulas (recall that we assumed $\mathcal{L}$ to be countable). If Γ is satisfiable, then it is satisfiable by an $\mathcal{L}$-structure with countable carrier set.*

Proof. We use the proof of the completeness theorem with $A = \bot$. It yields either $\Gamma \vdash_c \bot$ or else a model of $\Gamma \cup \{\neg\bot\}$ whose carrier set is the countable set $\mathsf{Ter}_{\mathcal{L}}$. But $\Gamma \vdash_c \bot$ cannot hold because of our assumption. □

Completeness for Minimal and Intuitionistic Logic

We now present a notion of a model suitable for minimal and intuitionistic logic. It is based on the idea (due to Beth [7]) of "developing possible worlds", indexed by elements k of a finitely branching tree. Knowledge can only increase, so if we have $R\boldsymbol{t}$ in one world k, then $R\boldsymbol{t}$ will also hold in future possible worlds.

Every Beth structure is based on a finitely branching tree T. Recall that a *node* over a nonempty set S is a finite sequence $k = \langle a_0, a_1, \dots, a_{n-1}\rangle$ of elements $a_i \in S$; n is the *length* $\mathsf{lh}(k)$ of k. We write $k \preceq k'$ if k is an initial segment of k'. A *tree* over S is a set of nodes (over S) closed under the formation of initial segments. A tree T is *finitely branching*, if every $k \in T$ has only finitely many immediate successors in T. A tree T is *unbounded* if for every $n \in \mathbb{N}$ there is a node $k \in T$ such that $\mathsf{lh}(k) = n$. A *branch* in a tree T is a linearly ordered (by $\preceq$) subtree of T.

For the completeness theorem it will suffice to consider Beth structures over the full binary tree, i.e. the set T_{01} of all finite 0-1-sequences (nodes) k. The empty sequence is denoted by $\langle\rangle$, and $k0$, $k1$ denote extensions of the sequence k by 0 or 1.

Definition 1.7.9. Let $(T, \preceq)$ be a finitely branching tree. $\mathcal{B} = (M, I_0, I_1)$ is an *$\mathcal{L}$-Beth structure* over T if (M, I_0) is an $\mathcal{L}$-prestructure and I_1 assigns to every n-ary relation symbol R of $\mathcal{L}$ and every node $k \in T$ an n-ary relation

$$I_1(R, k) \subseteq M^n$$

such that we have monotonicity, i.e.

$$k \preceq k' \implies I_1(R, k) \subseteq I_1(R, k').$$

If $n = 0$, then $I_1(R, k)$ is true or false and monotonicity says that for $k \preceq k'$, $I_1(R, k)$ implies $I_1(R, k')$.

Note that we do not require anything for $I_1(\bot, k)$; this corresponds to the fact that $\bot$ does not play a special role in minimal logic.

$t^{\mathcal{B}}[\eta]$ for an environment η is defined as for classical models. However, the model relation $\mathcal{M} \models A[\eta]$ there is replaced by the *forcing relation*.

Definition 1.7.10. $\mathcal{B}, k \Vdash A[\eta]$ ($\mathcal{B}$ at node k forces A for environment η) is defined inductively as follows. We write $k \Vdash A[\eta]$ when there is no doubt on the underlying structure $\mathcal{B}$, and $\forall k' \succeq_n k\, A$ for $\forall k' \succeq k . \mathsf{lh}(k') = \mathsf{lh}(k) + n \to A$.

1. $k \Vdash R(t_1, \dots, t_p)[\eta] \iff \exists n \forall k' \succeq_n k\, (t_1^{\mathcal{B}}[\eta], \dots, t_p^{\mathcal{B}}[\eta]) \in I_1(R, k')$ (or $I_1(R, k')$ is true, in case $p = 0$).
2. $k \Vdash (A \vee^* B)[\eta] \iff \exists n \forall k' \succeq_n k . k' \Vdash A[\eta]$ or $k' \Vdash B[\eta]$.
3. $k \Vdash \exists^* x A[\eta] \iff \exists n \forall k' \succeq_n k \exists a {\in} |\mathcal{B}|\, k' \Vdash A[\eta_x^a]$.
4. $k \Vdash (A \to B)[\eta] \iff \forall k' \succeq k . k' \Vdash A[\eta] \implies k' \Vdash B[\eta]$.
5. $k \Vdash (A \wedge B)[\eta] \iff k \Vdash A[\eta]$ and $k \Vdash B[\eta]$.
6. $k \Vdash (\forall x A)[\eta] \iff \forall a {\in} |\mathcal{B}|\, k \Vdash A[\eta_x^a]$.

So in the clauses for atoms, disjunctions and existential formulas we refer to a "bar" in the underlying finitely branching tree T. However, for atoms this is not necessary but just convenient when it comes to construct Beth structures. We can easily "complete" a given Beth structure by "pulling down the bar" for atoms. This means that we replace the originally given interpretation I_1 of the relation symbols by

$$\overline{I}_1(R, k) := \{\, \boldsymbol{a} \in |\mathcal{B}|^p \mid \exists n \forall k' \succeq_n k\, \boldsymbol{a} \in I_1(R, k') \,\}.$$

Then clearly we have monotonicity again, i.e. $\overline{I}_1(R, k) \subseteq \overline{I}_1(R, k')$ for $k \preceq k'$. In such a completed Beth structure we can simplify the atomic clause in the definition of forcing to

$$k \Vdash R\boldsymbol{t}[\eta] \iff \boldsymbol{t}^{\mathcal{B}}[\eta] \in \overline{I}_1(R, k).$$

From the definition it follows easily that monotonicity extends to formulas, i.e. that $k \Vdash A[\eta]$ implies $k' \Vdash A[\eta]$ for $k \preceq k'$. There is also a converse:

Lemma 1.7.11 (Covering Property).

$$\forall k' \succeq_n k\, k' \Vdash A[\eta] \implies k \Vdash A[\eta].$$

Proof. Induction on A. We write $k \Vdash A$ for $k \Vdash A[\eta]$.

Case $R\boldsymbol{t}$. Assume

$$\exists n \forall k' \succeq_n k\, k' \Vdash R\boldsymbol{t},$$

hence by definition

$$\exists n \forall k' \succeq_n k \exists m \forall k'' \succeq_m k'\, \boldsymbol{t}^{\mathcal{B}}[\eta] \in I_1(R, k'').$$

Since T is a finitely branching tree, we have

$$\exists m \forall k' \succeq_m k\, \boldsymbol{t}^{\mathcal{B}}[\eta] \in I_1(R, k'),$$

hence $k \Vdash R\boldsymbol{t}$.

The cases $A \vee^* B$ and $\exists^* x A$ are treated similarly.

Case $A \to B$. Assume $k' \Vdash A \to B$ for all $k' \succeq k$ such that $\mathsf{lh}(k') = \mathsf{lh}(k) + n$. We must show

$$\forall l \succeq k . l \Vdash A \implies l \Vdash B.$$

So let $l \succeq k$ and assume $l \Vdash A$. Our goal is $l \Vdash B$. We make use of the induction hypothesis for B with $m := \max(\mathsf{lh}(k) + n, \mathsf{lh}(l))$. So let $l' \succeq l$ and $\mathsf{lh}(l') = m$. It suffices to show $l' \Vdash B$. If $\mathsf{lh}(l') = \mathsf{lh}(l)$, then $l' = l$ and we are done. If $\mathsf{lh}(l') = \mathsf{lh}(k) + n > \mathsf{lh}(l)$, then l' is an extension of l and hence of k with length $\mathsf{lh}(k) + n$, and we have $l' \Vdash A \to B$ by assumption. Moreover $l' \Vdash A$, since $l' \succeq l$ and $l \Vdash A$. This again implies $l' \Vdash B$.

The cases $A \wedge B$ and $\forall x\, A$ are clear. □

The coincidence and substitution lemmas can be transferred in the expected way to Beth structures.

Lemma 1.7.12 (Coincidence). *Let $\mathcal{B}$ be a Beth structure, t a term, A a formula and η, ξ environments in $|\mathcal{B}|$.*

1. *If $\eta(x) = \xi(x)$ for all $x \in \mathsf{vars}(t)$, then $\eta(t) = \xi(t)$.*
2. *If $\eta(x) = \xi(x)$ for all $x \in \mathsf{FV}(A)$, then $\mathcal{B}, k \Vdash A[\eta] \iff \mathcal{B}, k \Vdash A[\xi]$.*

Proof. Induction on terms and formulas. □

Lemma 1.7.13 (Substitution). *Let $\mathcal{B}$ be a Beth structure, t, r $\mathcal{L}$-terms, A a formula and η an environment in $|\mathcal{B}|$. Then*

1. $\eta(r[x := t]) = \eta_x^{\eta(t)}(r)$.
2. $\mathcal{B}, k \Vdash A[x := t][\eta] \iff \mathcal{B}, k \Vdash A[\eta_x^{\eta(t)}]$.

Proof. Induction on terms and formulas. □

From these we obtain as usual the correctness theorem.

Theorem 1.7.14 (Correctness). *Let $\Gamma \cup \{A\}$ be a set of formulas. If $\Gamma \vdash A$, then $\Gamma \Vdash A$, i.e. for all Beth structures $\mathcal{B}$, nodes k and environments η*

$$\mathcal{B}, k \Vdash \Gamma[\eta] \implies \mathcal{B}, k \Vdash A[\eta].$$

Proof. Induction on derivations. *Case* $\forall^+$. Assume $k \Vdash \Gamma[\eta]$ and $x \notin \mathsf{FV}(\Gamma)$. We must show $k \Vdash \forall x A[\eta]$, i.e. $k \Vdash A[\eta_x^a]$ for an arbitrary $a \in |\mathcal{B}|$. But

$$\begin{array}{ll} k \Vdash \Gamma[\eta_x^a] & \text{by the coincidence lemma, for } x \notin \mathsf{FV}(\Gamma) \\ k \Vdash A[\eta_x^a] & \text{by induction hypothesis applied to } \Gamma \vdash A. \end{array}$$

Case $\forall^-$. Assume $k \Vdash \Gamma[\eta]$. We must show $k \Vdash A[x := t][\eta]$. Now

$k \Vdash \forall x A[\eta]$	by induction hypothesis applied to $\Gamma \vdash \forall x A$
$k \Vdash A[\eta_x^{\eta(t)}]$	by definition
$k \Vdash A[x := t][\eta]$	by the substitution lemma.

The other cases are left as an exercise. □

We now prove the converse.

Theorem 1.7.15 (Completeness). *Let $\Gamma \cup \{A\}$ be a set of formulas. The following are equivalent.*

1. *$\Gamma \vdash A$.*
2. *$\Gamma \Vdash A$, i.e. for all Beth structures $\mathcal{B}$, nodes k and environments η*

$$\mathcal{B}, k \Vdash \Gamma[\eta] \implies \mathcal{B}, k \Vdash A[\eta].$$

Proof. One direction is the correctness theorem. For the other direction we make use of an approach due to Friedman [13] and construct a Beth structure $\mathcal{B}$ (over the set T_{01} of all finite 0-1-sequences k ordered by the initial segment relation $k \preceq k'$) with the property that $\Gamma \vdash B$ is equivalent to $\mathcal{B}, \langle\rangle \Vdash B[\mathsf{id}]$.

To define $\mathcal{B}$ fix an enumeration $A_0, A_1, A_2, \dots$ of all $\mathcal{L}$-formulas, such that every formula appears infinitely often; also fix an enumeration $x_0, x_1, \dots$ of all variables. Let $\Gamma = \bigcup_n \Gamma_n$ with finite sets Γ_n such that $\Gamma_n \subseteq \Gamma_{n+1}$. To every node $k \in T_{01}$ we assign a finite set Δ_k of formulas, by induction on the length of k.

Let $\Delta_{\langle\rangle} := \emptyset$. Now assume that for a node k with $\mathsf{lh}(k) = n$ the set Δ_k is already defined. Let $\Delta \vdash_n B$ mean that there is a derivation of B from Δ with length (= total number of symbols) $\leq n$. We define Δ_{k0} and Δ_{k1} as follows.

Case 1. $\Gamma_n, \Delta_k \nvdash_n A_n$. Then let

$$\Delta_{k0} := \Delta_k \quad \text{and} \quad \Delta_{k1} := \Delta_k \cup \{A_n\}.$$

Case 2. $\Gamma_n, \Delta_k \vdash_n A_n = A'_n \vee^* A''_n$. Then let

$$\Delta_{k0} := \Delta_k \cup \{A_n, A'_n\} \quad \text{and} \quad \Delta_{k1} := \Delta_k \cup \{A_n, A''_n\}.$$

Case 3. $\Gamma_n, \Delta_k \vdash_n A_n = \exists^* x A'_n$. Then let

$$\Delta_{k0} := \Delta_{k1} := \Delta_k \cup \{A_n, A'_n[x := x_i]\},$$

with x_i the first variable $\notin \mathsf{FV}(\Gamma_n, A_n, \Delta_k)$.

Case 4. $\Gamma_n, \Delta_k \vdash_n A_n$, with A_n neither a disjunction nor an existential formula. Then let

$$\Delta_{k0} := \Delta_{k1} := \Delta_k \cup \{A_n\}.$$

Clearly $k \preceq k'$ implies $\Delta_k \subseteq \Delta_{k'}$. Note first that

$$\forall k' \succeq_n k \, \Gamma, \Delta_{k'} \vdash B \implies \Gamma, \Delta_k \vdash B. \tag{1.11}$$

To see this, it suffices to show

$$\Gamma, \Delta_{k0} \vdash B \quad \text{and} \quad \Gamma, \Delta_{k1} \vdash B \implies \Delta_k \vdash B.$$

This is clear in cases 1 and 4, and for cases 2 and 3 it follows easily from the schemes $\vee^{*-}$ and $\exists^{*-}$ (cf. section 1.6).

We now show that

$$\Gamma, \Delta_k \vdash B \implies \exists n \forall k' \succeq_n k \, B \in \Delta_{k'}. \tag{1.12}$$

To see this, choose $n \geq \mathsf{lh}(k)$ such that $B = A_n$ and $\Gamma_n, \Delta_k \vdash_n A_n$. Then for all $k' \succeq k$ with $\mathsf{lh}(k') = n+1$ we have $A_n \in \Delta_{k'}$ (cf. cases 2–4).

Using these sets Δ_k, we can now define our $\mathcal{L}$-Beth structure $\mathcal{B}$ to be $(\mathsf{Ter}_{\mathcal{L}}, I_0, I_1)$ with the canonical $I_0(f)\boldsymbol{t} := f\boldsymbol{t}$ and

$$\boldsymbol{t} \in I_1(R,k) \quad :\Longleftrightarrow \quad R\boldsymbol{t} \in \Delta_k.$$

Clearly $t^{\mathcal{B}}[\mathsf{id}] = t$ for all $\mathcal{L}$-terms t.

We finally show that

$$\Gamma, \Delta_k \vdash B \iff \mathcal{B}, k \Vdash B[\mathsf{id}], \tag{1.13}$$

by induction on the logical complexity of B. For $\mathcal{B}, k \Vdash B[\mathsf{id}]$ write $k \Vdash B$.

Case $R\boldsymbol{t}$. The following are equivalent.

$$\begin{array}{ll} \Gamma, \Delta_k \vdash R\boldsymbol{t} & \\ \exists n \forall k' {\succeq_n} k\, R\boldsymbol{t} \in \Delta_{k'} & \text{by (1.12) and (1.11)} \\ \exists n \forall k' {\succeq_n} k\, \boldsymbol{t} \in I_1(R,k') & \text{by definition of } \mathcal{B} \\ k \Vdash R\boldsymbol{t} & \text{by definition of } \Vdash \text{, since } t^{\mathcal{B}}[\mathsf{id}] = t. \end{array}$$

Case $B \vee^* C$. $\Longrightarrow$. Assume $\Gamma, \Delta_k \vdash B \vee^* C$. Choose $n \geq \mathsf{lh}(k)$ such that $\Gamma_n, \Delta_k \vdash_n A_n = B \vee^* C$. Then for all $k' \succeq k$ with $\mathsf{lh}(k') = n$ we have

$$\Delta_{k'0} := \Delta_{k'} \cup \{B \vee^* C, B\} \quad \text{and} \quad \Delta_{k'1} := \Delta_{k'} \cup \{B \vee^* C, C\},$$

so by induction hypothesis

$$k'0 \Vdash B \quad \text{and} \quad k'1 \Vdash C.$$

By definition this implies $k \Vdash B \vee^* C$. $\Longleftarrow$.

$$\begin{array}{ll} k \Vdash B \vee^* C & \\ \exists n \forall k' {\succeq_n} k\, .k' \Vdash B \ \text{ or } \ k' \Vdash C & \\ \exists n \forall k' {\succeq_n} k\, .\Gamma, \Delta_{k'} \vdash B \ \text{ or } \ \Gamma, \Delta_{k'} \vdash C & \text{by induction hypothesis} \\ \exists n \forall k' {\succeq_n} k\, \Gamma, \Delta_{k'} \vdash B \vee^* C & \\ \Gamma, \Delta_k \vdash B \vee^* C & \text{by (1.11).} \end{array}$$

The case $B \wedge C$ is obvious.

Case $B \to C$. $\Longrightarrow$. Assume $\Gamma, \Delta_k \vdash B \to C$. We have to show $k \Vdash B \to C$, i.e.

$$\forall k' {\succeq} k. k' \Vdash B \implies k' \Vdash C.$$

So let $k' \succeq k$ and assume $k' \Vdash B$. By induction hypothesis $\Gamma, \Delta_{k'} \vdash B$ and hence $\Gamma, \Delta_{k'} \vdash C$ by assumption. Again the induction hypothesis yields $k' \Vdash C$.

$\Longleftarrow$. Assume $k \Vdash B \to C$, i.e. $\forall k' {\succeq} k. k' \Vdash B \implies k' \Vdash C$. We have to show $\Gamma, \Delta_k \vdash B \to C$. To this end we will make use of (1.11). Pick $n \geq \mathsf{lh}(k)$ such that $B = A_n$. Let $k' \succeq_m k$ be arbitrary, where $m := n - \mathsf{lh}(k)$. We must show $\Gamma, \Delta_{k'} \vdash B \to C$.

In case $\Gamma, \Delta_{k'} \vdash_n A_n$ we have $k' \Vdash B$ by induction hypothesis, hence $k' \Vdash C$ by assumption, hence $\Gamma, \Delta_{k'} \vdash C$ again by induction hypothesis and therefore $\Gamma, \Delta_{k'} \vdash B \to C$.

In case $\Gamma, \Delta_{k'} \nvdash_n A_n$ by definition we have $\Delta_{k'1} = \Delta_{k'} \cup \{B\}$. This yields $\Gamma, \Delta_{k'1} \vdash B$, hence $k'1 \Vdash B$ by induction hypothesis, so $k'1 \Vdash C$ by assumption, hence $\Gamma, \Delta_{k'1} \vdash C$ again by induction hypothesis. Because of $\Delta_{k'1} = \Delta_{k'} \cup \{B\}$ also in this case we obtain $\Gamma, \Delta_{k'} \vdash B \to C$.

Case $\forall x B$. The following are equivalent.

$$\begin{array}{ll} \Gamma, \Delta_k \vdash \forall x B & \\ \forall t {\in} \mathsf{Ter}_{\mathcal{L}}\, \Gamma, \Delta_k \vdash B[x := t] & \\ \forall t {\in} \mathsf{Ter}_{\mathcal{L}}\, k \Vdash B[x := t] & \text{by induction hypothesis} \\ \forall t {\in} \mathsf{Ter}_{\mathcal{L}}\, k \Vdash B[\mathsf{id}_x^t] & \text{by the substitution lemma, since } t^{\mathcal{B}}[\mathsf{id}] = t, \\ k \Vdash \forall x B & \text{by definition of } \Vdash. \end{array}$$

Case $\exists^* xB$. Similarly to case $\vee^*$. In detail, the proof is as follows. $\Longrightarrow$. Assume $\Gamma, \Delta_k \vdash \exists^* xB$. Choose $n \geq \mathsf{lh}(k)$ such that $\Gamma_n, \Delta_k \vdash_n A_n = \exists^* xB$. Then for all $k' \succeq k$ with $\mathsf{lh}(k') = n$ we have

$$\Delta_{k'0} = \Delta_{k'1} = \Delta_k \cup \{\exists^* xB, B[x := x_i]\}$$

with x_i not free in $\Delta_k \cup \{\exists^* xB\}$, so by induction hypothesis

$$k'0 \Vdash B[x := x_i] \quad \text{and} \quad k'1 \Vdash B[x := x_i].$$

By definition this implies $k \Vdash \exists^* xB$. $\Longleftarrow$.

$$\begin{array}{ll}
k \Vdash \exists^* xB & \\
\exists n \forall k' \succeq_n k \exists t \in \mathsf{Ter}_{\mathcal{L}}\, k' \Vdash B[\mathsf{id}_x^t] & \\
\exists n \forall k' \succeq_n k \exists t \in \mathsf{Ter}_{\mathcal{L}}\, k' \Vdash B[x := t] & \\
\exists n \forall k' \succeq_n k \exists t \in \mathsf{Ter}_{\mathcal{L}}\, \Gamma, \Delta_{k'} \vdash B[x := t] & \text{by induction hypothesis} \\
\exists n \forall k' \succeq_n k\, \Gamma, \Delta_{k'} \vdash \exists^* xB & \\
\Gamma, \Delta_k \vdash \exists^* xB & \text{by (1.11).}
\end{array}$$

Now we can finish the proof of the completeness theorem. By (1.13) we have $\mathcal{B}, \langle\rangle \Vdash \Gamma[\mathsf{id}]$. By assumption this implies $\mathcal{B}, \langle\rangle \Vdash A[\mathsf{id}]$, hence $\Gamma \vdash A$ again by (1.13). □

An an immediate corollary we obtain completeness for intuitionistic logic.

Corollary 1.7.16. *Let $\Gamma \cup \{A\}$ be a set of formulas. The following are equivalent.*

1. *$\Gamma \vdash_i A$.*
2. *$\Gamma, \mathsf{Efq} \Vdash A$, i.e. for all Beth structures $\mathcal{B}$ with $\mathcal{B} \Vdash \mathsf{Efq}$, nodes k and environments η*

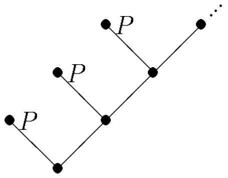
$$\mathcal{B}, k \Vdash \Gamma[\eta] \implies \mathcal{B}, k \Vdash A[\eta].$$ □

Countermodels

The completeness theorem makes it easy to construct counterexamples for intuitionistic derivability. First note that if $\perp$ is never forced, then

$$\begin{aligned}
k \Vdash \neg A &\iff \forall k' \succeq k\, k' \not\Vdash A, \\
k \Vdash \neg\neg A &\iff \forall k' \succeq k\, k' \not\Vdash \neg A \\
&\iff \forall k' \succeq k \exists k'' \succeq k'\, k'' \Vdash A.
\end{aligned}$$

Example 1. $\not\vdash_i \neg\neg P \to P$. To describe a Beth structure, we indicate the underlying finitely branching tree as a diagram, writing next to each node the propositional variables forced at that node. Consider the Beth structure given by



Clearly we have

$$\begin{aligned}
&\langle\rangle \not\Vdash P, \\
&\langle\rangle \Vdash \neg\neg P.
\end{aligned}$$

Hence $\langle\rangle \not\Vdash \neg\neg P \to P$ and therefore $\not\vdash \neg\neg P \to P$. Since clearly $\Vdash \mathsf{Efq}_R$ for every R, we also have $\not\vdash_i \neg\neg P \to P$. The model also shows underivability of the Peirce formula $((P \to Q) \to P) \to P$ in intuitionistic logic.

Example 2. $\nvdash_i \forall x \neg\neg Px \to \neg\neg\forall x Px$. We construct a countermodel over the full binary tree T_{01} with domain $\mathbb{N}$. Let

$$n \in I_1(P, k) \quad :\Longleftrightarrow \quad \text{there are} > n \text{ ones in } k.$$

We first show $\forall k\, k \nVdash \forall x Px$ (which implies $\langle\rangle \Vdash \neg\forall x Px$, hence $\langle\rangle \nVdash \neg\neg\forall x Px$). Let k be given, and let n denote the number of ones in k. Then at all nodes $k00\dots0$ we do not have Pn, i.e. $k \nVdash \forall x Px$.

We now show $\langle\rangle \Vdash \neg\neg Pn$, for an arbitrary n (which implies $\langle\rangle \Vdash \forall x \neg\neg x Px$). It suffices to prove $\forall k \exists k' {\succeq} k\; k' \Vdash Pn$; let $k' := k11\dots1$ with n appended ones.

Kripke structures

Kripke structures have been introduced in [21] as an alternative to Beth structures. They are simpler in the sense that the clauses for $\vee^*, \exists^*$ need not refer to a "bar" but only to the node at hand. However, they are more complex in the sense that they do not have a constant domain any more, but the domains increase with increasing node k.

Definition 1.7.17. Let $(T, \preceq)$ be a finitely branching tree. $\mathcal{K} = (D, I_0, I_1)$ is an $\mathcal{L}$-*Kripke structure* if D assigns to every node $k \in T$ a non-empty set $D(k)$ such that $k \preceq k'$ implies that $(D(k), I_0(k))$ is an $\mathcal{L}$-sub-prestructure of $(D(k'), I_0(k'))$ and I_1 assigns to every n-ary relation symbol R of $\mathcal{L}$ and every node $k \in T$ an n-ary relation

$$I_1(R, k) \subseteq D(k)^n$$

such that we have monotonicity, i.e.

$$k \preceq k' \implies I_1(R, k) \subseteq I_1(R, k').$$

If $n = 0$, then $I_1(R, k)$ is true or false and monotonicity says that for $k \preceq k'$, $I_1(R, k)$ implies $I_1(R, k')$. $\mathcal{K}, k \Vdash A[\eta]$ ($\mathcal{K}$ at node k forces A for environment η) is defined inductively as follows. We write $k \Vdash A[\eta]$ when $\mathcal{K}$ is clear from the context.

1. $k \Vdash R\boldsymbol{t} \iff \boldsymbol{t}^{\mathcal{K},k}[\eta] \in I_1(R, k)$ (or $I_1(R, k)$ is true, in case $m = 0$).
2. $k \Vdash (A \vee^* B)[\eta] \iff k \Vdash A[\eta]$ or $k \Vdash B[\eta]$.
3. $k \Vdash \exists^* x A[\eta] \iff \exists a{\in}D(k)\; k \Vdash A[\eta_x^a]$.
4. $k \Vdash (A \to B)[\eta] \iff \forall k'{\succeq}k. k' \Vdash A[\eta] \implies k' \Vdash B[\eta]$.
5. $k \Vdash (A \wedge B)[\eta] \iff k \Vdash A[\eta]$ and $k \Vdash B[\eta]$.
6. $k \Vdash (\forall x A)[\eta] \iff \forall k'{\succeq}k \forall a{\in}D(k')\; k' \Vdash A[\eta_x^a]$.

The completion of Beth structures makes comparison to Kripke structures easier. A completed Beth structure is a Kripke structure with constant domains, and both notions of forcing coincide for formulas without $\vee^*, \exists^*$. But note that Kripke structures are somewhat more flexible when it comes to construct (finite) countermodels, because not every Kripke structure with constant domains is a completed Beth structure. An obvious example is

However, Kripke structures are less general than Beth structures, in the sense that any Kripke structure can in a suitable sense be "transformed" into a Beth structure with constant domain $\mathbb{N}$; this was observed by Kripke in [21]. We now describe this transformation, following Troelstra and van Dalen [30].

Let $\mathcal{K} = (D, I_0, I_1)$ be a Kripke structure whose underlying finitely branching tree is $(T, \preceq)$. We transform $\mathcal{K}$ into a Beth structure $\mathcal{B}$ over the finitely branching tree T' consisting of all finite nondecreasing sequences over $(T, \preceq)$ of length > 0 and ordered by $\sigma \preceq' \tau$ iff σ is an initial segment of τ.

Split $\mathbb{N}$ into countably many disjoint sets N_i, i.e. $\mathbb{N} = \bigcup_{i \in \mathbb{N}} N_i$ and $i \neq j \to N_i \cap N_j = \emptyset$, and let $M_i := \bigcup_{j \leq i} N_i$. Clearly we can define for every $\sigma = \langle k_0, \dots, k_n \rangle \in T'$ a surjection ψ_σ from M_n onto $D(k_n)$ such that

$\psi_{\sigma k} a = \psi_\sigma a$ for $a \in M_{\mathsf{lh}(\sigma)-1}$,
$\psi_{\sigma k}$ maps $M_{\mathsf{lh}(\sigma)}$ onto $D(k)$.

We define our new Beth structure as follows. Let $\sigma = \langle k_0, \dots, k_n \rangle$.

$$(a_1, \dots, a_p) \in I_1(R, \sigma) \quad :\Longleftrightarrow \quad a_1, \dots, a_p \in M_n \text{ and } k_n \Vdash R(\psi_\sigma a_1, \dots, \psi_\sigma a_p)$$

(or more precisely, $k_n \Vdash R(x_1, \dots, x_p)[\psi_\sigma a_1/x_1, \dots, \psi_\sigma a_p/x_p]$; generally, we write $k \Vdash A(a_1, \dots, a_p)$ for $k \Vdash A[\psi_\sigma a_1/x_1, \dots, \psi_\sigma a_p/x_p]$, provided $x_1, \dots, x_p$ contain the free variables of A). This clearly defines a Beth structure $\mathcal{B}$, whose forcing relation is denoted by $\Vdash'$.

Lemma 1.7.18. *Let $\sigma = \langle k_0, \dots, k_n \rangle$. Then for all $a_1, \dots, a_p \in M_n$*

$$\sigma \Vdash' A(a_1, \dots, a_p) \iff k_n \Vdash A(\psi_\sigma a_1, \dots, \psi_\sigma a_p).$$

Proof. By induction on A. *Case* $\forall x B$. $\Longrightarrow$. Assume $\sigma \Vdash' \forall x B(x, a')$ with $a' \in M_n$. We must show $k_n \Vdash \forall x B(x, \psi_\sigma a')$. So let $a \in D(k_n)$ be given; we must show $k_n \Vdash B(a, \psi_\sigma a')$. Since $a = \psi_\sigma b$ for some $b \in M_n$, this means $k_n \Vdash B(\psi_\sigma b, \psi_\sigma a')$, so by induction hypothesis it suffices to show $\sigma \Vdash' B(b, a')$. But this follows from our assumption.

$\Longleftarrow$. Assume $k_n \Vdash \forall x B(x, \psi_\sigma a')$ with $a' \in M_n$. We must show $\sigma \Vdash' \forall x B(x, a')$. So let $a \in \mathbb{N}$ be given; we then must show $\sigma \Vdash' B(a, a')$. Recall that $\sigma = \langle k_0, \dots, k_n \rangle$. Choose $\tau = \sigma * \langle k_{n+1}, \dots, k_{n+i}, k' \rangle$ with $k_n \preceq k'$ long enough such that $a \in M_{\mathsf{lh}(\tau)-1}$. Recall that ψ_τ maps $M_{\mathsf{lh}(\tau)-1}$ onto $D(k')$ and $\psi_\tau a' = \psi_\sigma a' \in D(k_n)$. From $k_n \Vdash \forall x B(x, \psi_\sigma a')$ and $k_n \preceq k'$ we obtain $k' \Vdash \forall x B(x, \psi_\tau a')$, hence $k' \Vdash B(\psi_\tau a, \psi_\tau a')$, hence by induction hypothesis $\tau \Vdash' B(a, a')$. This holds for all continuations τ of σ with $a \in M_{\mathsf{lh}(\tau)-1}$ (i.e., with at least this length). Hence by the covering property 1.7.11 we obtain $\sigma \Vdash' B(a, a')$, as required.

The other cases are easier and left as an exercise. □

1.8 Classical Completeness from Completeness of Minimal Logic

We present a proof of the completeness theorem for classical logic from the completeness theorem for minimal logic.

Theorem 1.8.1 (Completeness). *Let $\Gamma \cup \{A\}$ be a set of formulas. The following are equivalent.*

1. *$\Gamma \vdash_c A$.*
2. *$\Gamma \models A$, i.e. for all structures $\mathcal{M}$ and environments η*

$$\mathcal{M} \models \Gamma[\eta] \implies \mathcal{M} \models A[\eta].$$

Proof. One direction is the correctness theorem. For the other direction we make use of the completeness theorem for minimal logic.

Clearly it suffices to only consider formulas without $\vee^*, \exists^*$, and (by lemma 1.7.3) also without $\wedge$.

Assume $\Gamma \nvdash_c A$, i.e. $\Gamma, \mathsf{Stab} \nvdash A$. By the completeness theorem for minimal logic we have a countable Beth structure $\mathcal{B} = (\mathsf{Ter}_\mathcal{L}, I_0, I_1)$ over the full binary tree T_{01} and a node l_0 such that $l_0 \Vdash \Gamma, \mathsf{Stab}$ and $l_0 \nVdash A$ (writing $k \Vdash B$ for $\mathcal{B}, k \Vdash B[\mathsf{id}]$).

Call a node k *consistent* if $k \nVdash \bot$, and *stable* if $k \Vdash \mathsf{Stab}$. Let k be a stable node and B a formula (without $\vee^*, \exists^*$). We have stability $\mathsf{Stab} \vdash \neg\neg B \to B$ by lemma 1.4.2, hence $k \Vdash \neg\neg B \to B$, hence

$$\begin{aligned} k \nVdash B &\iff k \nVdash \neg\neg B \\ &\iff \exists k' {\succeq} k.\, k' \text{ consistent and } k' \Vdash \neg B. \end{aligned} \tag{1.14}$$

Let α be a branch in the underlying tree T_{01}. We define

$$\begin{aligned} \alpha \Vdash A \quad &:\Longleftrightarrow \quad \exists k{\in}\alpha\, k \Vdash A, \\ \alpha \text{ is consistent} \quad &:\Longleftrightarrow \quad \alpha \nVdash \bot, \\ \alpha \text{ is stable} \quad &:\Longleftrightarrow \quad \exists k{\in}\alpha\, k \Vdash \mathsf{Stab}. \end{aligned}$$

Note that

$$\text{If } \alpha \Vdash \boldsymbol{A} \text{ and } \vdash \boldsymbol{A} \to B, \text{ then } \alpha \Vdash B. \tag{1.15}$$

To see this, assume $\alpha \Vdash \boldsymbol{A}$. Then $k \Vdash \boldsymbol{A}$ for some $k \in \alpha$, since α is linearly ordered. Because of $\vdash \boldsymbol{A} \to B$ correctness yields $k \Vdash B$, i.e. $\alpha \Vdash B$.

Call a branch α *generic* (in the sense that it generates a classical model) if it is consistent and stable, for all formulas B

$$\alpha \Vdash B \text{ or } \alpha \Vdash \neg B, \tag{1.16}$$

and for all formulas $\forall \boldsymbol{y} B$ (where $\boldsymbol{y}$ is nonempty) with B not an all-formula

$$\forall \boldsymbol{s} {\in} \mathsf{Ter}_{\mathcal{L}}\, \alpha \Vdash B[\boldsymbol{y} := \boldsymbol{s}] \implies \alpha \Vdash \forall \boldsymbol{y} B \tag{1.17}$$

Finally we define for a branch α a classical structure $\mathcal{M}^\alpha = (\mathsf{Ter}_{\mathcal{L}}, I_0, I_1^\alpha)$ by

$$I_1^\alpha(R) := \bigcup_{k \in \alpha} I_1(R, k) \quad \text{for } R \neq \bot.$$

We show that for every generic branch α and formula B in the language based on $\{\to, \forall\}$ we have

$$\alpha \Vdash B \iff \mathcal{M}^\alpha \models B. \tag{1.18}$$

The proof is by induction on the logical complexity of B.

Case $R\boldsymbol{t}$, $R \neq \bot$. The claim then holds for any α.

Case $\bot$. We have $\alpha \nVdash \bot$, for α is consistent.

Case $B \to C$. $\Longrightarrow$. Assume $\alpha \Vdash B \to C$ and $\mathcal{M}^\alpha \models B$. We must show $\mathcal{M}^\alpha \models C$. Now $\alpha \Vdash B$ by induction hypothesis, hence $\alpha \Vdash C$, hence $\mathcal{M}^\alpha \models C$ again by induction hypothesis. $\Longleftarrow$. Assume $\mathcal{M}^\alpha \models B \to C$. If $\mathcal{M}^\alpha \models B$, then $\mathcal{M}^\alpha \models C$, hence $\alpha \Vdash C$ by induction hypothesis and therefore $\alpha \Vdash B \to C$. If $\mathcal{M}^\alpha \not\models B$, then $\alpha \nVdash B$ by induction hypothesis, hence $\alpha \Vdash \neg B$ by (1.16) and therefore $\alpha \Vdash B \to C$ since α is stable (and $\vdash (\neg\neg C \to C) \to \bot \to C$).

Case $\forall \boldsymbol{y} B$ (where $\boldsymbol{y}$ is nonempty) with B not an all-formula. The following are equivalent.

$$\begin{array}{ll} \alpha \Vdash \forall \boldsymbol{y} B & \\ \forall \boldsymbol{s} {\in} \mathsf{Ter}_{\mathcal{L}}\, \alpha \Vdash B[\boldsymbol{y} := \boldsymbol{s}] & \text{by (1.17)} \\ \forall \boldsymbol{s} {\in} \mathsf{Ter}_{\mathcal{L}}\, \mathcal{M}^\alpha \models B[\boldsymbol{y} := \boldsymbol{s}] & \text{by induction hypothesis} \\ \mathcal{M}^\alpha \models \forall \boldsymbol{y} B. & \end{array}$$

We finally show that for every consistent stable node k, there is a generic branch containing k. For the proof let $A_0, A_1, \dots$ be an enumeration of all formulas. We inductively define an infinite sequence $k = k_0 \preceq k_1 \preceq k_2 \dots$ of consistent stable nodes. So let $k_0 := k$. Now assume that k_n already is constructed. Write A_n in the form $\forall \boldsymbol{y} B$ (where $\boldsymbol{y}$ may be empty) with B not an all-formula. If $k_n \Vdash \forall \boldsymbol{y} B$, let $k_{n+1} := k_n$. Otherwise $k_n \nVdash B[\boldsymbol{y} := \boldsymbol{s}]$ for some $\boldsymbol{s}$, and by (1.14) there is a consistent node $k' \succeq k_n$ such that $k' \Vdash \neg B[\boldsymbol{y} := \boldsymbol{s}]$. Let $k_{n+1} := k'$. Because of $k_n \preceq k_{n+1}$ also k_{n+1} is stable.

Let $\alpha := \{\, l \mid \exists n\, l \preceq k_n \,\}$, hence $k \in \alpha$. We now show that α is generic. Clearly α is consistent and stable. Both (1.16) and (1.17) can be proved simultaneously. Let $C = \forall \boldsymbol{y} B$ with B not an all-formula, and choose n such that $C = A_n$. If $k_n \Vdash \forall \boldsymbol{y} B$ we are done. Otherwise $k_n \nVdash B[\boldsymbol{y} := \boldsymbol{s}]$ for some $\boldsymbol{s}$, and by construction $k_{n+1} \Vdash \neg B[\boldsymbol{y} := \boldsymbol{s}]$. For (1.16) we obtain $k_{n+1} \Vdash \neg\forall \boldsymbol{y} B$ (since $\vdash \forall \boldsymbol{y} B \to B[\boldsymbol{y} := \boldsymbol{s}]$), and for (1.17) we have a contradiction to the consistency of α.

Now we can finish the proof of the completeness theorem. Since $l_0 \nVdash A$ and l_0 is stable, (1.14) yields a consistent $k \succeq l_0$ such that $k \Vdash \neg A$. Clearly also k is stable. Now by what we just proved there is a generic branch α with $k \in \alpha$. Because of $k \Vdash \neg A$ we have $\alpha \Vdash \neg A$, hence $\mathcal{M}^\alpha \models \neg A$ by (1.18). Moreover $\alpha \Vdash \Gamma$, hence $\mathcal{M}^\alpha \models \Gamma$ again by (1.18). Therefore $\Gamma \not\models A$. $\square$

1.9 Decision Procedure for Intuitionistic Propositional Logic

We present work of Weich [31], for the implicational fragment of intuitionistic propositional logic. It is shown that searching simultaneously for a proof and a countermodel gives an efficient decision algorithm. This work may be seen as the result of looking for the computational content (rather than the operational semantics) of the rules in Hudelmaier's calculus [19].

Weich introduced the novel rule $\to$L below, and moreover a calculus which selects (similar to PROLOG) only those clauses whose head is the goal atom (or more precisely, allows passage to the goal atom via simple implications $P_1 \to P_2$ only). Then it is possible to simultaneously construct a derivation or a countermodel, where the depth of the search tree is bounded by the number of implications in the formula.

Every formula in the language based on $\to$ can be written uniqely in the form $A_1 \to A_2 \to \cdots \to A_n \to P$ with an atom P; for better readability we write this in sequent form $\Gamma \Rightarrow P$ with $\Gamma = A_1, A_2, \ldots, A_n$ a list of formulas.

A *root countermodel* to $\Gamma \Rightarrow P$ is a Kripke structure $\mathcal{K}$ such that $\mathcal{K} \Vdash \Gamma$ and $\mathcal{K} \nVdash P$. A formula D is a *properly nested implication* if $D = (\boldsymbol{A}_1 \to Q_1) \to \cdots \to (\boldsymbol{A}_n \to Q_n) \to Q$ where $n > 1$, or $n = 1$ and $\boldsymbol{A}_1$ not empty.

The central idea is to consider the following derived rule.

$$\frac{\boldsymbol{A}_1, \Gamma_1, Q_1 \to Q, \Gamma_2 \Rightarrow Q_1 \qquad \boldsymbol{A}_n, \Gamma_1, Q_n \to Q, \Gamma_2 \Rightarrow Q_n}{\Gamma_1, (\boldsymbol{A}_1 \to Q_1) \to \cdots \to (\boldsymbol{A}_n \to Q_n) \to Q, \Gamma_2 \Rightarrow Q} \to\mathsf{L}$$

Note that for $n > 1$, or $n = 1$ and $\boldsymbol{A}_1$ not empty, each premise has fewer occurrences of $\to$ than the conclusion.

To see that this is a derived rule we argue informally. To simplify notation let $n = 2$, $\boldsymbol{A_1} = A_1$, $\boldsymbol{A_2} = A_2$ and Γ_1, Γ_2 be empty. So assume

$$\begin{aligned} &u_1 \colon A_1 \to (Q_1 \to Q) \to Q_1, \\ &u_2 \colon A_2 \to (Q_2 \to Q) \to Q_2, \\ &u_3 \colon (A_1 \to Q_1) \to (A_2 \to Q_2) \to Q. \end{aligned}$$

We must show Q. By u_3 it suffices to show $A_1 \to Q_1$ and $A_2 \to Q_2$.

For $A_1 \to Q_1$: Assume A_1. We must show Q_1. By u_1 it suffices to show $Q_1 \to Q$. So assume Q_1. We must show Q. By u_3 it suffices to show $A_2 \to Q_2$. So assume A_2. We must show Q_2. By u_2 it suffices to show $Q_2 \to Q$. So assume Q_2. We must show Q. But this follows from u_3.

For $A_2 \to Q_2$: Assume A_2. We must show Q_2. By u_2 it suffices to show $Q_2 \to Q$. So assume Q_2. We must show Q. By u_3 it suffices to show $A_1 \to Q_1$. So assume A_1. We must show Q_1. By u_1 it suffices to show $Q_1 \to Q$. So assume Q_1. We must show Q. But this follows from u_3.

Implications which are not properly nested, i.e. of the form $P_1 \to P_2$, are easy to treat directly. A *chain* in Γ from P_1 to P_n is a list of implications $P_1 \to P_2$, $P_2 \to P_3$, $\ldots$, $P_{n-1} \to P_n$ in Γ. An atomic formula P_1 is *significant* for $\Gamma \Rightarrow P$ if there is a chain from P_1 to P in Γ.

Theorem 1.9.1. *Every sequent $\Gamma \Rightarrow P$ has a derivation or a countermodel.*

Proof. By induction on the number of implications in Γ. Let significant mean significant for $\Gamma \Rightarrow P$.

Case 1. There is a significant Q in Γ. Then by definition there is a chain from Q to P in Γ, and this clearly yields a derivation of $\Gamma \Rightarrow P$.

Case 2. For a partition $\Gamma = \Gamma_1, D, \Gamma_2$ with D a properly nested implication $D = (\boldsymbol{A}_1 \to Q_1) \to \cdots \to (\boldsymbol{A}_n \to Q_n) \to Q$ and Q significant, the induction hypothesis on $\boldsymbol{A}_i, \Gamma_1, Q_i \to Q, \Gamma_2 \Rightarrow Q_i$ yields a derivation for each $1 \leq i \leq n$. From all these derivations we obtain a derivation of $\Gamma \Rightarrow Q$ and hence (since Q is significant) also a derivation of $\Gamma \Rightarrow P$.

Case 3. There is no significant atom in Γ, and for every partition $\Gamma = \Gamma_1, D, \Gamma_2$ with D a properly nested implication $D = (\boldsymbol{A}_1 \to Q_1) \to \cdots \to (\boldsymbol{A}_n \to Q_n) \to Q$ and Q significant, the induction hypothesis yields a countermodel $\mathcal{K}_i$ to $\boldsymbol{A}_i, \Gamma_1, Q_i \to Q, \Gamma_2 \Rightarrow Q_i$. Clearly $\mathcal{K}_i$ is also a countermodel to $\boldsymbol{A_i}, \Gamma \Rightarrow Q_i$.

Let L be the set of all atomic formulas significant for $\Gamma \Rightarrow P$, and $\mathcal{K}$ the Kripke structure built from the $\boldsymbol{\mathcal{K}}$, whose root forces all atoms forced in every $\mathcal{K}_i$ and which are not significant for $\Gamma \Rightarrow P$, i.e. not in L. We show that $\mathcal{K}$ is a countermodel to $\Gamma \Rightarrow P$.

We have $\mathcal{K} \nVdash P$, since $P \in L$. So we must show $\mathcal{K} \Vdash \Gamma$. Let $D \in \Gamma$; we must show $\mathcal{K} \Vdash D$. By assumption $\mathcal{K}_i$ is a countermodel to $\boldsymbol{A_i}, \Gamma \Rightarrow Q_i$, hence $\mathcal{K}_i \Vdash \Gamma$, hence $\mathcal{K}_i \Vdash D$ for all i.

Subcase D is atomic. By assumption $D \notin L$, and $\mathcal{K}_i \Vdash D$ for all i. By definition of $\mathcal{K}$ this means $\mathcal{K} \Vdash D$.

Subcase $D = P_1 \to P_2$. Since $\mathcal{K}_i \Vdash D$, it suffices to show that $\mathcal{K} \Vdash P_1$ implies $\mathcal{K} \Vdash P_2$. Assume $\mathcal{K} \Vdash P_1$, i.e. $P_1 \notin L$ and $\mathcal{K}_i \Vdash P_1$ for all i. Because of $\mathcal{K}_i \Vdash P_1 \to P_2$ we have $\mathcal{K}_i \Vdash P_2$ for all i . Now $P_1 \to P_2 \in \Gamma$ and $P_1 \notin L$, hence $P_2 \notin L$. So $\mathcal{K} \Vdash P_2$.

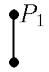
Subcase $D = (\boldsymbol{A}_1 \to Q_1) \to \cdots \to (\boldsymbol{A}_n \to Q_n) \to Q$ is a properly nested implication. Since $\mathcal{K}_i \Vdash D$, it suffices to show that if $\mathcal{K} \Vdash \boldsymbol{A}_i \to Q_i$ for all i, then $\mathcal{K} \Vdash Q$.

Subsubcase Q is significant. Then $\mathcal{K} \nVdash \boldsymbol{A}_i \to Q_i$, because of $\mathcal{K} \preceq \mathcal{K}_i$.

Subsubcase Q is not significant, i.e. $Q \notin L$. By construction of $\mathcal{K}$ it suffices to show that $\mathcal{K}_i \Vdash Q$ for every i. Since $\mathcal{K}_i \Vdash D$, for this it suffices to show that $\mathcal{K}_i \Vdash \boldsymbol{A}_j \to Q_j$ for all j. But this follows via monotonicity from the assumption that $\mathcal{K} \Vdash \boldsymbol{A}_j \to Q_j$ for all j. □

The proof describes a decision algorithm for the implicational fragment of minimal propositional logic. Here are some examples.

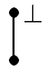
1. Consider the Peirce formula $((P_1 \to P_2) \to P_1) \to P_1$. The induction hypothesis yields a root countermodel to $P_1, P_2 \to P_1 \Rightarrow P_2$, consisting of a single node where all non-significant atoms hold, i.e. in our case just P_1. The resulting root countermodel for the Peirce formula is obtained by putting a root node below, where at the root all significant atoms (for the present sequent $(P_1 \to P_2) \to P_1 \Rightarrow P_1$) are removed; in our case this is P_1, so the root node forces no atom. The final root countermodel is



2. The very same countermodel is obtained for $\bot \to P_1, \bot \to P_2, (P_1 \to P_2) \to P_1 \Rightarrow P_1$. This shows intuitionistic underivability of the Peirce formula.

3. Similarly we can treat $((P \to \bot) \to \bot) \to P$.

4. Somewhat more interesting is the formula $((((P \to \bot) \to \bot) \to P) \to \bot) \to \bot$, which is intuitionistically provable by Glivenko's theorem (see problem 1.11.6). The induction hypothesis yields a root countermodel to $\neg\neg P, \neg P \Rightarrow P$, consisting of a single node where all non-significant atoms hold, i.e. in our case just $\bot$. The resulting root countermodel for the final formula is obtained by putting a root node below, where at the root all significant atoms (for the final formula) are removed; in our case this is $\bot$, so the root node forces no atom. The final root countermodel is



1.10 Notes

Beth structures for intuitionistic logic have been introduced in [7]; however, the (classical and intuitionistic) completeness proofs given there were in need of correction. He later revised his work in [8]. One of his intentions was to give *intuitionistic* completeness proofs for intuitionistic logic. This aspect has been particularly stressed by Kreisel; it is ignored in the present exposition. A good survey on intuitionistic completeness proofs can be found in Troelstra's book [29]. Kripke semantics for intuitionistic logic was developed in [21], and a (classical) Gödel-style completeness proof was given. The construction of a Beth structure from a Kripke structure is due to Kripke [21]; the present exposition benefitted from Troelstra and van Dalen [30].

The proof in section 1.8 of the completeness theorem for classical logic from the completeness theorem for minimal logic is due to Berger.

Work related to that of Weich discussed in section 1.9 has been done by Underwood. Starting from Fitting's [12] completeness proof she outlined how a proof of decidability could be implemented in a formal system like NUPRL. Underwood's extracted program has a sequent as input and returns a derivation or a countermodel. Her program uses a loop-check and tries to construct bottom-up a repetition-free derivation in a sequent calculus, similarly to the decision procedure already given by Gentzen [15]. To avoid a loop-check, contraction-free calculi were introduced by Hudelmaier [18] and Dyckhoff [10] rediscovering the work of Vorob'ev (Amer Math Soc Transl 94 (1970)). Formalizing their proof would be a quite hard task. Fortunately, Miglioli, Moscato and Ornaghi [25] gave an alternative completeness proof of a similar system by constructing a Kripke structure, and this provided the inspiration for Weich's paper. Hudelmaier developed his calculus further and gave an $O(n \log n)$-space decision algorithm [19].

1.11 Problems

Problem 1.11.1. Show that the following schemes are derivable in minimal logic.

$$(A \vee^* B \to C) \leftrightarrow (A \to C) \wedge (B \to C), \tag{1.19}$$
$$(A \wedge B \to C) \leftarrow (A \to C) \vee^* (B \to C), \tag{1.20}$$
$$(A \to B \wedge C) \leftrightarrow (A \to C) \wedge (B \to C), \tag{1.21}$$
$$(A \to B \vee^* C) \leftarrow (A \to C) \vee^* (B \to C). \tag{1.22}$$

$$A \wedge \forall x B \leftrightarrow \forall x. A \wedge B \qquad \text{if } x \notin \mathsf{FV}(A), \tag{1.23}$$
$$A \wedge \exists^* x B \leftrightarrow \exists^* x. A \wedge B \qquad \text{if } x \notin \mathsf{FV}(A), \tag{1.24}$$
$$A \vee^* \forall x B \to \forall x. A \vee^* B \qquad \text{if } x \notin \mathsf{FV}(A), \tag{1.25}$$
$$A \vee^* \exists^* x B \leftrightarrow \exists^* x. A \vee^* B \qquad \text{if } x \notin \mathsf{FV}(A). \tag{1.26}$$

$$\forall x A \wedge \forall x B \leftrightarrow \forall x. A \wedge B, \tag{1.27}$$
$$\forall x A \vee^* \forall x B \to \forall x. A \vee^* B, \tag{1.28}$$
$$\exists^* x A \wedge \exists^* x B \leftarrow \exists^* x. A \wedge B, \tag{1.29}$$
$$\exists^* x A \vee^* \exists^* x B \leftrightarrow \exists^* x. A \vee^* B. \tag{1.30}$$

$$(A \to \forall x B) \leftrightarrow \forall x. A \to B \qquad \text{if } x \notin \mathsf{FV}(A), \tag{1.31}$$
$$(A \to \exists^* x B) \leftarrow \exists^* x. A \to B \qquad \text{if } x \notin \mathsf{FV}(A), \tag{1.32}$$
$$(\forall x A \to B) \leftarrow \exists^* x. A \to B \qquad \text{if } x \notin \mathsf{FV}(B), \tag{1.33}$$
$$(\exists^* x A \to B) \leftrightarrow \forall x. A \to B \qquad \text{if } x \notin \mathsf{FV}(B). \tag{1.34}$$

Problem 1.11.2. Show that

a. $\vdash A \to \neg\neg A$
b. $\vdash \neg\neg\neg A \leftrightarrow \neg A$
c. $\vdash \neg\neg(A \wedge B) \leftrightarrow (\neg\neg A \wedge \neg\neg B)$
d. $\vdash \neg\neg(A \to B) \to (\neg\neg A \to \neg\neg B)$
e. $\vdash \neg\neg\forall x A \to \forall x \neg\neg A$

Problem 1.11.3. Show that $\vdash_i A \vee^* \neg A \to \neg\neg A \to A$.

Problem 1.11.4. Show that $\vdash (\bot \to B) \to (A \to \neg\neg B) \to \neg\neg(A \to B)$. Note that $\vdash (\bot \to B) \to (\neg\neg A \to \neg\neg B) \to \neg\neg(A \to B)$ follows, because the hypothesis of a hypothesis is weakened (A to $\neg\neg A$).

Problem 1.11.5. Show that

$$\vdash_c \neg(B \to C) \to B \wedge \neg C,$$
$$\vdash_c (\neg B \to A) \wedge (C \to A) \to (B \to C) \to A,$$
$$\vdash_c (\forall x B \to A) \to \forall y(\neg B[x := y] \to A) \to A \quad \text{for } y \text{ not free in } \forall x B,\ A.$$

Problem 1.11.6. (Glivenko [17]).

a. $\vdash_c A \iff \vdash_i \neg\neg A$ for A quantifier-free. (Hint: use (1.19)).

b. Let DNS ("double negation shift") be the schema $\forall x \neg\neg A \to \neg\neg\forall x A$. Show that $\vdash_c B \iff \mathrm{DNS} \vdash_i \neg\neg B$. (Hint: use (1.34)).

Problem 1.11.7. Prove by means of countermodels that

a. $\nvdash_i \neg\neg(\forall x \neg\neg A \to \neg\neg\forall x A)$.
b. $\nvdash_i \neg\neg\exists^* x A \to \exists x \neg\neg A$.

Problem 1.11.8. Show that

$$\vdash ((A \to C) \to A) \to ((B \to C) \to B) \to (A \to B \to C) \to C.$$

2. Proof Terms in Typed Lambda Calculus; Normalization

The first goal of this chapter is to introduce a compact and convenient notation for derivations, which essentially replaces the two-dimensional tree-like notation used up to now by a linear one. To take care of the possibility to introduce and later to cancel assumptions, we need free (assumption) variables and also a device to bind them, traditionally called λ. This immediately leads to typed λ-calculi.

We also show in this chapter that every derivation can be transformed by appropriate conversion steps into a normal form. A derivation in normal form has the property that it does not make "detours", or more precisely, that an elimination rule never immediately follows an introduction rule. Derivations in normal form have many pleasant properties, and can be used for a variety of results.

Uniqueness of normal form will be shown by means of an application of Newman's lemma; we will also introduce and discuss the related notions of confluence, local confluence and the Church-Rosser property.

This is first done for the $\to$-fragment of propositional logic, since nearly all essential features already come up in this special case. The extension to the full language is discussed afterwards.

As applications we prove the subformula property and Herbrand's theorem. We finally show that the requirement to give a normal derivation of a derivable formula can sometimes be unrealistic. Following Statman [27] and Orevkov [24] we give examples of formulas C_k which are easily derivable with non-normal derivations (whose number of nodes is linear in k), but which require a non-elementary (in k) number of nodes in any normal derivation.

2.1 Simply Typed λ-Calculus

We begin by considering the fragment of minimal logic dealing only with formulas built from propositional symbols using the connective $\to$, i.e. the $\to$-fragment of minimal propositional logic. Derivations are built using introduction and elimination rules for $\to$. They have a particularly simple structure, since it is not necessary to take the eigenvariable condition into account (because there are no quantifiers). Note however that assumption variables may be bound.

The central idea for our term notation for derivations is to consider *formulas as types* of derivation terms. This concept is known as the *Curry-Howard correspondence*. Formulas of the $\to$-fragment of minimal propositional logic are now called (simple) *types*, denoted by ρ, σ, τ. Special types are the propositional symbols (and $\bot$), which in this context are called *ground types*; they are denoted by ι. So

- Every ground type ι is a type.
- If ρ and σ are types, then so is $\rho \to \sigma$.

Observe that every type ρ can be written uniquely in the form $\rho = \rho_1 \to \rho_2 \to \ldots \to \rho_n \to \iota$; the ρ_i are called *argument types* of ρ.

λ-terms with types correspond to derivations in the $\to$-fragment of minimal propositional logic. They are denoted M, N, K, L, and we shall usually signify their type by writing it as a superscript thus M^ρ, N^σ etcetera. Sometimes M^ρ is written instead as $M\colon \rho$. The assumption variables $\Box_0, \Box_1, \Box_2, \ldots$ are now called simply *variables*, denoted by x, y, z.

Definition 2.1.1 (Inductive definition of λ-terms with types).
- Every variable x^ρ is a λ-term.
- If M^σ is a λ-term, then so is $(\lambda x^\rho M^\sigma)^{\rho\to\sigma}$ (*abstraction*).
- If $M^{\rho\to\sigma}$ and N^ρ are λ-terms, then so is $(M^{\rho\to\sigma} N^\rho)^\sigma$ (*application*).

λ-terms will often be simply called terms, and type indices will be omitted whenever they are clear from the context or inessential. Let Λ (Λ_ρ) denote the set of all terms (of type ρ). Terms not of the form of an abstraction are called *neutral*. For application terms we adopt the convention that they should associate to the left, so MNK means $(MN)K$, not $M(NK)$. We again sometimes use a dot notation if the range of a λ-operator should be as large as admitted by the surrounding parentheses, so e.g. $(\lambda x.MN)K$ means $(\lambda x(MN))K$. The set $\mathsf{FV}(M)$ of variables free in the λ-term M is defined recursively by

$$\begin{aligned} \mathsf{FV}(x) &:= \{x\} \\ \mathsf{FV}(\lambda x M) &:= \mathsf{FV}(M) \setminus \{x\} \\ \mathsf{FV}(MN) &:= \mathsf{FV}(M) \cup \mathsf{FV}(N) \end{aligned}$$

Extension to Conjunction and the Universal Quantifier

The extension to the case of propositional logic including conjunction $\wedge$, i.e. to product types, is rather straightforward; instead of $\rho \wedge \sigma$ we write $\rho \times \sigma$. The inductive definition of λ-terms with types is extended by the clauses

- If M^ρ and N^σ are λ-terms, then so is $\langle M^\rho N^\sigma \rangle^{\rho\times\sigma}$ (*pairing*).
- If $M^{\rho\times\sigma}$ is a λ-term, then so are $\pi_0(M)^\rho$ and $\pi_1(M)^\sigma$ (*projections*).

Sometimes it will be convenient to write $M0$ for $\pi_0(M)$ and $M1$ for $\pi_1(M)$.

We may as well extend the term representation to our full $\wedge\to\forall$-language. This creates no esssential new difficulties, only some care in the notations is needed. *The main difference in notation will now be that the formulas themselves will be used as type symbols, rather than ρ, σ etcetera as before.* Often we also write $u\colon A$ instead of u^A and $M\colon A$ instead of M^A.

As building blocks for our terms denoting derivations we use assumption variables decorated with a formula u^A; here u again is one of the symbols $\Box_0, \Box_1, \dots$. As before we require that different assumption formulas have different markers u. Derivation terms are again denoted by M, N, K, L and assumption variables by u, v, w.

We give an inductive definition of derivation terms in table 2.1, where for clarity we have written the corresponding derivations to the left.

To formulate the variable condition we need the set $\mathsf{FA}(M)$ of assumption variables free in the derivation term M. It is defined recursively by

$$\begin{aligned} \mathsf{FA}(u^A) &:= \{u^A\}, \\ \mathsf{FA}(\langle M^A, N^B \rangle^{A\wedge B}) &:= \mathsf{FA}(M^A) \cup \mathsf{FA}(N^B), \\ \mathsf{FA}(\pi_0(M^{A\wedge B})^A) &:= \mathsf{FA}(M^{A\wedge B}), \\ \mathsf{FA}(\pi_1(M^{A\wedge B})^B) &:= \mathsf{FA}(M^{A\wedge B}), \\ \mathsf{FA}((\lambda u^A M^B)^{A\to B}) &:= \mathsf{FA}(M^B) \setminus \{u^A\}, \\ \mathsf{FA}((M^{A\to B} N^A)^B) &:= \mathsf{FA}(M^{A\to B}) \cup \mathsf{FA}(N^A), \\ \mathsf{FA}((\lambda x M^A)^{\forall x A}) &:= \mathsf{FA}(M^A), \\ \mathsf{FA}((M^{\forall x A} t)^{A[x:=t]}) &:= \mathsf{FA}(M^{\forall x A}). \end{aligned}$$

The variable condition then is that $x \notin \mathsf{FV}(A)$ for all $u^A \in \mathsf{FA}(M)$.

Note that formula superscripts in derivation terms are often superfluous. For instance, in $(\lambda u M)^{A\to B}$ it is clear that the assumption variable u must carry the formula A and the derivation term M must carry the formula B as superscript. Similarly in $(M^{A\to B} N)$ the whole term must have B and N must have A as superscript. In what follows, we will often omit such redundant formulas. This will provide us with a very compact notation for derivations.

Examples for Derivation Terms

We now write down the example derivations (1.1)-(1.4) as derivation terms.

derivation	term
$u\colon A$	u^A
$\dfrac{\begin{matrix} \mid M \\ A \end{matrix} \quad \begin{matrix} \mid N \\ B \end{matrix}}{A \wedge B}\,\wedge^+$	$\langle M^A, N^B \rangle^{A\wedge B}$
$\dfrac{\begin{matrix} \mid M \\ A \wedge B \end{matrix}}{A}\,\wedge_0^- \qquad \dfrac{\begin{matrix} \mid M \\ A \wedge B \end{matrix}}{B}\,\wedge_1^-$	$\pi_0(M^{A\wedge B})^A \qquad \pi_1(M^{A\wedge B})^B$
$\dfrac{\begin{matrix} [u\colon A] \\ \mid M \\ B \end{matrix}}{A \to B}\,\to^+ u$	$(\lambda u^A M^B)^{A\to B}$
$\dfrac{\begin{matrix} \mid M \\ A \to B \end{matrix} \quad \begin{matrix} \mid N \\ A \end{matrix}}{B}\,\to^-$	$(M^{A\to B} N^A)^B$
$\dfrac{\begin{matrix} \mid M \\ A \end{matrix}}{\forall x A}\,\forall^+ x$ (with var.cond.)	$(\lambda x M^A)^{\forall x A}$ (with var.cond.)
$\dfrac{\begin{matrix} \mid M \\ \forall x A \end{matrix} \quad t}{A[x := t]}\,\forall^-$	$(M^{\forall x A} t)^{A[x:=t]}$

Table 2.1. The Curry-Howard correspondence between natural deduction and typed λ calculus

(1.1) With $D := (A \wedge B \to C) \to A \to B \to C$ we have

$$\left\{ \lambda u^{A\wedge B \to C} \left(\lambda v^A \left[\lambda w^B \left(u^{A\wedge B\to C} \langle v^A, w^B \rangle^{A\wedge B} \right)^C \right]^{B\to C} \right)^{A\to B\to C} \right\}^D$$

It is clear that many redundant formulas may be omitted here. With assumption variables u, v, w of types $u\colon A \wedge B \to C$, $v\colon A$ and $w\colon B$ we write the derivation term shortly as

$$\lambda u \lambda v \lambda w (u \langle vw \rangle).$$

(1.2) We use assumption variables $u\colon A \to (B \to C)$ and $v\colon A \wedge B$. The derivation term then is

$$\lambda u \lambda v [u(\pi_0 v)](\pi_1 v).$$

(1.3) With $u\colon \forall x.A \to B$ and $v\colon A$ we have

$$\lambda u \lambda v \lambda x [(ux)v]$$

(1.4) With $u\colon A \to \forall x B$ and $v\colon A$ we have

$$\lambda u \lambda x \lambda v [(uv)x].$$

A derivation term M^A is called *closed*, if $\mathsf{FA}(M^A) = \emptyset$. We write

$$M^B[u_1^{A_1}, \dots, u_n^{A_n}]$$

to express that the assumption variables free in M^B are contained in the list $u_1^{A_1}, \dots, u_n^{A_n}$.

We will also need the set $\mathsf{FV}(M)$ of free object variables in a derivation term M. It is defined by

$$\begin{aligned}
\mathsf{FV}(u^A) &:= \mathsf{FV}(A),\\
\mathsf{FV}(\langle M^A, N^B \rangle) &:= \mathsf{FV}(M^A) \cup \mathsf{FV}(N^B),\\
\mathsf{FV}(\pi_i(M^{A\wedge B})) &:= \mathsf{FV}(M^{A\wedge B}),\\
\mathsf{FV}(\lambda u^B M^A) &:= \mathsf{FV}(M^A),\\
\mathsf{FV}(M^{A\to B} N^A) &:= \mathsf{FV}(M^{A\to B}) \cup \mathsf{FV}(N^A),\\
\mathsf{FV}(\lambda x M^A) &:= \mathsf{FV}(M^A) \setminus \{x\},\\
\mathsf{FV}(M^{\forall x A} t) &:= \mathsf{FV}(M^{\forall x A}) \cup \mathsf{vars}(t).
\end{aligned}$$

For example, consider the following derivation terms.

(1) $M_1 = \lambda v^P u^{Qx}$.

$$\dfrac{u\colon Qx}{P \to Qx}\ \to^+ v$$

Then we have $\mathsf{FA}(M_1) = \{u^{Qx}\}$ and $\mathsf{FV}(M_1) = \{x\}$.

(2) $M_2 = \lambda u^{Qx} \lambda v^P u^{Qx}$.

$$\dfrac{\dfrac{u\colon Qx}{P \to Qx}\ \to^+ v}{Qx \to (P \to Qx)}\ \to^+ u$$

Then we have $\mathsf{FA}(M_2) = \emptyset$ and $\mathsf{FV}(M_2) = \{x\}$.

(3) $M_3 = \lambda x \lambda u^{Qx} \lambda v^P u^{Qx}$.

$$\dfrac{\dfrac{\dfrac{u\colon Qx}{P \to Qx}\ \to^+ v}{Qx \to (P \to Qx)}\ \to^+ u}{\forall x. Qx \to (P \to Qx)}\ \forall^+ x$$

Then we have $\mathsf{FA}(M_3) = \emptyset$ and $\mathsf{FV}(M_3) = \emptyset$.

Note that for derivation terms we have two kinds of substitution: one can substitute a derivation term M^A for an assumption variable u^A, and also an object term t for an object variable x.

Constants for $\exists^*$

The axioms for the strong existential quantifier $\exists^*$ are denoted by appropriate constants:

$$\exists^{*+}_{x,A}\colon \forall x.A \to \exists^* x A$$
$$\exists^{*-}_{x,A,B}\colon \exists^* x A \to (\forall x.A \to B) \to B$$

with the usual proviso $x \notin \mathsf{FV}(B)$.

2.2 Normalization for the $\to$-Fragment of Minimal Propositional Logic

For what follows it will be often helpful to make use of a second (and clearly equivalent) inductive definition of terms, which does not explicitely contain an application case:

$$M ::= x\boldsymbol{M} \mid \lambda x M \mid (\lambda x M)N\boldsymbol{L},$$

where $\boldsymbol{M}$ denotes a finite list $M_1, \ldots, M_n$ of terms. By a *conversion rule* we mean a relation between terms of the same type. In particular, we want to study the rules of *β-conversion* and *η-conversion*. They are defined by

$$(\lambda x M)N\boldsymbol{L} \mapsto_\beta M[x := N]\boldsymbol{L},$$
$$\lambda x.Mx \mapsto_\eta M \quad \text{if } x \notin \mathsf{FV}(M) \text{ and } M \text{ is neutral.}$$

A term M is called *β-convertible* if it has the form of a left hand side of $\mapsto_\beta$. Any such term is called a *β-redex* (for "reducible expression"). Correspondingly we define η-convertible terms and η-redexes.

We now give an inductive definition of the *term closure* of a conversion rule $\mapsto_\gamma$.

Definition 2.2.1. – If $M \mapsto_\gamma M'$, then $M \longrightarrow_\gamma M'$ (*head conversion*).
– If $M \longrightarrow_\gamma M'$, then also

$$x\boldsymbol{M}M\boldsymbol{N} \longrightarrow_\gamma x\boldsymbol{M}M'\boldsymbol{N},$$
$$(\lambda x M)\boldsymbol{N} \longrightarrow_\gamma (\lambda x M')\boldsymbol{N},$$
$$(\lambda x N)\boldsymbol{M}M\boldsymbol{N} \longrightarrow_\gamma (\lambda x N)\boldsymbol{M}M'\boldsymbol{N}.$$

These reductions are called *inner reductions*.

In particular we have now defined *β-reduction* $\longrightarrow_\beta$ and *η-reduction* $\longrightarrow_\eta$. *$\beta\eta$-reduction* is defined by $\longrightarrow_{\beta\eta} := \longrightarrow_\beta \cup \longrightarrow_\eta$.

Note that β-conversion can produce new β-redexes. Examples are

$$(\lambda x x)(\lambda y M)N \longrightarrow_\beta (\lambda y M)N \longrightarrow_\beta M[x := N],$$
$$(\lambda x \lambda y M)NK \longrightarrow_\beta (\lambda y M[x := N])K \longrightarrow_\beta M[x, y := N, K],$$
$$(\lambda x.xM)\lambda y N \longrightarrow_\beta (\lambda y N)M[x := \lambda y N] \longrightarrow_\beta N[y := M[x := \lambda y N]].$$

In *type free λ-calculus* a special case of the last example yields the loop

$$(\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta (\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta (\lambda x.xx)(\lambda x.xx) \longrightarrow_\beta \ldots.$$

In the λ-calculus with types the term $\lambda x.xx$ cannot be formed. We will see soon that generally such infinite reduction sequences cannot occur.

Let $\longrightarrow$ be a binary relation on a non empty set U; we call $(U, \longrightarrow)$ a *reduction system*. We define the relations $\longrightarrow^i$ (i-fold product), $\longrightarrow^+$ (transitive closure) and $\longrightarrow^*$ (reflexive transitive closure) by

$$x \longrightarrow^0 y \;:\Longleftrightarrow\; x = y,$$
$$x \longrightarrow^{i+1} y \;:\Longleftrightarrow\; \exists z. x \longrightarrow^i z \wedge z \longrightarrow y,$$
$$x \longrightarrow^+ y \;:\Longleftrightarrow\; \exists i{>}0 (x \longrightarrow^i y),$$
$$x \longrightarrow^* y \;:\Longleftrightarrow\; \exists i{\geq}0 (x \longrightarrow^i y).$$

We also write $x \xrightarrow{*} y$ for $x \longrightarrow^* y$ and $x \xrightarrow{i} y$ for $x \longrightarrow^i y$. The relations $\longleftarrow$ and $\longleftrightarrow$ are defined by

$$\begin{aligned} x \longleftarrow y \quad &:\Longleftrightarrow \quad y \longrightarrow x, \\ x \longleftrightarrow y \quad &:\Longleftrightarrow \quad x \longrightarrow y \text{ or } y \longrightarrow x. \end{aligned}$$

Hence $\longleftrightarrow$ is the symmetric closure of $\longrightarrow$, and $\xleftrightarrow{*}$ is the reflexive symmetric transitive closure of $\longrightarrow$, i.e. the least equivalence relation containing $\longrightarrow$.

Lemma 2.2.2. *For $\longrightarrow$ being either of $\longrightarrow_\beta$ or $\longrightarrow_{\beta\eta}$ we have the following substitution properties.*

1. *If $M \longrightarrow M'$, then $MN \longrightarrow M'N$.*
2. *If $N \longrightarrow N'$, then $MN \longrightarrow MN'$.*
3. *If $M \longrightarrow M'$, then $M[x := N] \longrightarrow M'[x := N]$.*
4. *If $N \longrightarrow N'$, then $M[x := N] \xrightarrow{*} M[x := N']$.*

Proof. 1. By cases according to the generation of $M \longrightarrow M'$. For $\longrightarrow_\beta$ the claim is obvious; for $\longrightarrow_{\beta\eta}$ note that in case of an η head conversion $\lambda x.Mx \mapsto_\eta M$ we have $(\lambda x.Mx)N \mapsto_\beta MN$.

2. By cases on M.

3. Induction on $M \longrightarrow M'$. *Case* $\mapsto_\beta$. Clearly $\mapsto_\beta$ is closed under substitution. *Case* $\lambda y.My \mapsto_\eta M$ with M neutral, $y \notin \mathsf{FV}(M)$. If $M[x := N]$ is neutral, then $\lambda y.M[x := N]y \mapsto_\eta M[x := N]$. Otherwise $M[x := N] = \lambda yK$, and $\lambda y.(\lambda yK)y \longrightarrow_\beta \lambda yK$. *Case* $y\boldsymbol{M}M\boldsymbol{N} \longrightarrow y\boldsymbol{M}M'\boldsymbol{N}$. For $y \neq x$ the claim follows from the induction hypothesis. Now let $y = x$. We must show that $N\boldsymbol{M}[x := N]M[x := N]\boldsymbol{N}[x := N] \longrightarrow N\boldsymbol{M}[x := N]M'[x := N]\boldsymbol{N}[x := N]$. By induction hypothesis $M[x := N] \longrightarrow M'[x := N]$. Using (1) and (2) we obtain the claim. In the other cases the claim also follows easily from the induction hypothesis.

4. Induction on M. *Case* x. Obvious. *Case* $y \neq x$. Obvious. *Case* MK. Then $M[x := N]K[x := N] \xrightarrow{*} M[x := N']K[x := N] \xrightarrow{*} M[x := N']K[x := N']$ by induction hypothesis, (1) and (2). *Case* λyM. Then $\lambda yM[x := N] \xrightarrow{*} \lambda yM[x := N']$ by induction hypothesis. □

We now prove termination of $\longrightarrow_{\beta\eta}$, by which we mean that every reduction sequence terminates in a finite number of steps. Let $\gamma \in \{\beta, \eta, \beta\eta\}$. A term M is called *γ-normal* (or in *γ normal form*), if there is no M' such that $M \longrightarrow_\gamma M'$. By a *reduction sequence* we mean a finite or infinite sequence $M_1, M_2, \dots, M_n, \dots$ of terms such that $M_i \longrightarrow_\gamma M_{i+1}$. A term M is called *strongly γ-normalizable* if every γ reduction sequence starting with M terminates, i.e. stops after finitely many steps. β-normal terms are clearly characterized by the inductive definition

$$M ::= x\boldsymbol{M} \mid \lambda xM.$$

$\beta\eta$-normal terms are characterized by the same same rules, where however the last is to be restricted on terms M not of the form Nx with $x \notin \mathsf{FV}(N)$.

We want to show that every term M is strongly $\beta\eta$-normalizable, i.e. that the relation $\longrightarrow_{\beta\eta}$ on Λ is terminating. In particular then also $\longrightarrow_\beta$ is terminating.

We define by recursion on k a relation $\mathsf{sn}(M, k)$ between terms M and natural numbers k with the intention that k is an upper bound on the number of reduction steps up to normal form.

Definition 2.2.3.

$$\begin{aligned} \mathsf{sn}(M, 0) \quad &\Longleftrightarrow \quad M \text{ is in normal form,} \\ \mathsf{sn}(M, k+1) \quad &\Longleftrightarrow \quad \text{for all } M' \text{ such that } M \longrightarrow_{\beta\eta} M' \text{ we have } \mathsf{sn}(M', k). \end{aligned}$$

Clearly a term is strongly normalizable if there is a k such that $\mathsf{sn}(M, k)$. We first prove some closure properties of the relation sn.

Lemma 2.2.4. *1. If $\mathsf{sn}(M, k)$, then $\mathsf{sn}(M, k+1)$.*

2. *If $\mathsf{sn}(MN, k)$, then $\mathsf{sn}(M, k)$.*
3. *If $\mathsf{sn}(M_i, k_i)$ for $i = 1 \dots n$, then $\mathsf{sn}(xM_1 \dots M_n, k)$ with $k := k_1 + \dots + k_n$.*
4. *If $\mathsf{sn}(M, k)$, then $\mathsf{sn}(\lambda xM, k+1)$.*
5. *If $\mathsf{sn}(M[x := N]\boldsymbol{L}, k)$ and $\mathsf{sn}(N, l)$, then $\mathsf{sn}((\lambda xM)N\boldsymbol{L}, k+l+1)$.*

Proof. We write $\longrightarrow$ for $\longrightarrow_{\beta\eta}$.

1. Induction on k. Assume $\mathsf{sn}(M,k)$. We have to show $\mathsf{sn}(M,k+1)$. So let M' with $M \longrightarrow M'$ be given; because of $\mathsf{sn}(M,k)$ we must have $k>0$. We have to show $\mathsf{sn}(M',k)$. Because of $\mathsf{sn}(M,k)$ we have $\mathsf{sn}(M',k-1)$, hence by induction hypothesis $\mathsf{sn}(M',k)$.

2. Induction on k. Assume $\mathsf{sn}(MN,k)$. We have to show $\mathsf{sn}(M,k)$. In case $k=0$ the term MN is normal, hence also M is normal and therefore $\mathsf{sn}(M,0)$. Now let $k>0$ and $M \longrightarrow M'$; we have to show $\mathsf{sn}(M',k-1)$. From $M \longrightarrow M'$ we have $MN \longrightarrow M'N$. Because of $\mathsf{sn}(MN,k)$ we have by definition $\mathsf{sn}(M'N,k-1)$. The induction hypothesis now yields $\mathsf{sn}(M',k-1)$.

3. Assume $\mathsf{sn}(M_i,k_i)$ for $i=1\dots n$. We have to show $\mathsf{sn}(xM_1\dots M_n,k)$ with $k := k_1+\dots+k_n$. Again we employ induction on k. In case $k=0$ all M_i are normal, hence also $xM_1\dots M_n$. Now let $k>0$ and $xM_1\dots M_n \longrightarrow M'$. Then $M' = xM_1\dots M_i'\dots M_n$ with $M_i \longrightarrow M_i'$; we have to show $\mathsf{sn}(xM_1\dots M_i'\dots M_n, k-1)$. Because of $M_i \longrightarrow M_i'$ and $\mathsf{sn}(M_i,k_i)$ we have $k_i>0$ and $\mathsf{sn}(M_i',k_i-1)$. The induction hypothesis now yields $\mathsf{sn}(xM_1\dots M_i'\dots M_n,k-1)$.

4. Assume $\mathsf{sn}(M,k)$. We have to show $\mathsf{sn}(\lambda xM,k+1)$. Use induction on k. In case $k=0$ M is normal. If $M = Nx$ with $x \notin \mathsf{FV}(N)$, then there is exactly one term L such that $\lambda x.Nx \longrightarrow L$, i.e. N, and N is normal. Hence we have $\mathsf{sn}(\lambda x.Nx,1)$. If M is not of the form Nx with $x \notin \mathsf{FV}(N)$, then λxM is normal, hence $\mathsf{sn}(\lambda xM,1)$ by (1). Now let $k>0$ and $\lambda xM \longrightarrow L$. We have to show $\mathsf{sn}(L,k)$.

Case $M = Nx$ with $x \notin \mathsf{FV}(N)$ and $L = N$, i.e. we have a head η conversion. From $\mathsf{sn}(Nx,k)$ we can infer $\mathsf{sn}(N,k)$ by (2).

Case $L = \lambda xM'$ with $M \longrightarrow M'$. From $\mathsf{sn}(M,k)$ we can conclude $k>0$ and $\mathsf{sn}(M',k-1)$. The induction hypothesis now yields $\mathsf{sn}(\lambda xM',k-1+1)$.

5. Assume $\mathsf{sn}(M[x:=N]\boldsymbol{L},k)$ and $\mathsf{sn}(N,l)$. We need to show that $\mathsf{sn}((\lambda xM)N\boldsymbol{L},k+l+1)$. We use induction on $k+l$. In case $k+l=0$ the term N and $M[x:=N]\boldsymbol{L}$ are normal, hence also M and all L_i. Hence there is exactly one term K such that $(\lambda xM)N\boldsymbol{L} \longrightarrow K$, i.e. $M[x:=N]\boldsymbol{L}$, and this K is normal. Therefore we have $\mathsf{sn}((\lambda xM)N\boldsymbol{L},1)$. Now let $k+l>0$ and $(\lambda xM)N\boldsymbol{L} \longrightarrow K$. We have to show $\mathsf{sn}(K,k+l)$.

Case $K = M[x:=N]\boldsymbol{L}$, i.e. we have a head conversion. From $\mathsf{sn}(M[x:=N]\boldsymbol{L},k)$ we obtain $\mathsf{sn}(M[x:=N]\boldsymbol{L},k+l)$ by (1).

Case $K = (\lambda xM')N\boldsymbol{L}$ with $M \longrightarrow M'$. Then we have $M[x:=N]\boldsymbol{L} \longrightarrow M'[x:=N]\boldsymbol{L}$. Now $\mathsf{sn}(M[x:=N]\boldsymbol{L},k)$ implies $k>0$ and $\mathsf{sn}(M'[x:=N]\boldsymbol{L},k-1)$. The induction hypothesis yields $\mathsf{sn}((\lambda xM')N\boldsymbol{L},k-1+l+1)$.

Case $K = (\lambda xM)N'\boldsymbol{L}$ with $N \longrightarrow N'$. Now $\mathsf{sn}(N,l)$ implies $l>0$ and $\mathsf{sn}(N',l-1)$. The induction hypothesis yields $\mathsf{sn}((\lambda xM)N'\boldsymbol{L},k+l-1+1)$, since $\mathsf{sn}(M[x:=N']\boldsymbol{L},k)$ by (1),

Case $K = (\lambda xM)N\boldsymbol{L}'$ with $L_i \longrightarrow L_i'$ for some i and $L_j = L_j'$ for $j \neq i$. Then we have $M[x:=N]\boldsymbol{L} \longrightarrow M[x:=N]\boldsymbol{L}'$. Now $\mathsf{sn}(M[x:=N]\boldsymbol{L},k)$ implies $k>0$ and $\mathsf{sn}(M[x:=N]\boldsymbol{L}',k-1)$. The induction hypothesis yields $\mathsf{sn}((\lambda xM)N\boldsymbol{L}',k-1+l+1)$. □

The essential idea of the proof now is to view the last three closure properties of sn from the preceeding lemma without the information on the bounds as an inductive definition of a new set SN.

Definition 2.2.5. 1. If $\boldsymbol{M} \in \mathsf{SN}$, then $x\boldsymbol{M} \in \mathsf{SN}$.
2. If $M \in \mathsf{SN}$, then $\lambda xM \in \mathsf{SN}$.
3. If $M[x:=N]\boldsymbol{L} \in \mathsf{SN}$ and $N \in \mathsf{SN}$, then $(\lambda xM)N\boldsymbol{L} \in \mathsf{SN}$.

Corollary 2.2.6. *For every term $M \in \mathsf{SN}$ there is a k such that $\mathsf{sn}(M,k)$. Hence every term $M \in \mathsf{SN}$ is strongly normalizable.*

Proof. By induction on $M \in \mathsf{SN}$, using lemma 2.2.4 □

In what follows we shall show that *every* term is in SN and hence is strongly normalizable. Given the definition of SN we only have to show that SN is closed under application. In order to prove this we must prove simultaneously closure of SN under substitution.

Lemma 2.2.7. *For all types ρ, all $M \in \mathsf{SN}$ and all $N^\rho \in \mathsf{SN}$ we have*

1. *$M[x:=N] \in \mathsf{SN}$.*
2. *Suppose M has a type $\rho \to \sigma$. Then $MN \in \mathsf{SN}$.*

Proof. Induction on ρ and side induction on $M \in \mathsf{SN}$.

Case $y\boldsymbol{M}$. Let $N^\rho \in \mathsf{SN}$. 1. The SIH(1) (side induction hypothesis (1)) yields $(M_i)[x := N] \in \mathsf{SN}$ for all M_i from $\boldsymbol{M}$. In case $x \neq y$ we immediately have $(y\boldsymbol{M})[x := N] \in \mathsf{SN}$. Otherwise we need $N\boldsymbol{M}[x := N] \in \mathsf{SN}$. But this follows by multiple applications of IH(2) (induction hypothesis (2)), since every $(M_i)[x := N]$ has an argument type of ρ as its type. 2. $y\boldsymbol{M}N \in \mathsf{SN}$ follows from the definition of SN.

Case λyM. Let again $N^\rho \in \mathsf{SN}$. 1. By SIH(1) we know $M[x := N] \in \mathsf{SN}$, hence $\lambda yM[x := N] \in \mathsf{SN}$ by definition of SN. 2. By definition of SN it suffices to show for $(\lambda yM)N \in \mathsf{SN}$, that $M[x := N] \in \mathsf{SN}$ and $N \in \mathsf{SN}$. The latter holds by assumption, and the former by SIH(1).

Case $(\lambda yM)K\boldsymbol{L}$. Let again $N^\rho \in \mathsf{SN}$. 1. The SIH(1) for $M[y := K]\boldsymbol{L}$ and K yields $M[x := N][y := K[x := N]]\boldsymbol{L}[x := N] \in \mathsf{SN}$ and $K[x := N] \in \mathsf{SN}$, hence $(\lambda yM[x := N])K[x := N]\boldsymbol{L}[x := N] \in \mathsf{SN}$ by definition of SN. 2. The SIH(2) for $M[y := K]\boldsymbol{L}$ yields $M[y := K]\boldsymbol{L}N \in \mathsf{SN}$, hence $(\lambda yM)K\boldsymbol{L}N \in \mathsf{SN}$ by definition of SN. □

Corollary 2.2.8. *For every term we have $M \in \mathsf{SN}$; in particular every term M is strongly normalizable.*

Proof. Induction on the (first) inductive definition of terms M. In cases x and λxM the claim follows from the definition of SN, and in case MN it follows from part (2) of the preceeding lemma. □

2.3 Newman's Lemma; Uniqueness of Normal Form

The arguments in this section hold for an arbitrary binary relation $\longrightarrow$ on a non empty set U; we call $(U, \longrightarrow)$ a *reduction system*.

We are interested in the case $U = \Lambda$ and $\longrightarrow \in \{\longrightarrow_\beta, \longrightarrow_\eta, \longrightarrow_{\beta\eta}\}$, where we have just seen that $\longrightarrow$ is terminating, and hence that every $u \in U$ has at least one normal form. Now we look for conditions under which any u has at most one normal form. This property is called *confluence*. Will also need two further properties, *local confluence* and the *Church-Rosser property*.

Local confluence is weaker than confluence, but usually much easier to check. However, it is a very useful property, since for a terminating reduction system we will prove that local confluence implies confluence. The design of the Church-Rosser property is motivated by the desire to solve the so-called *word problem* by means of reduction techniques. The word problem for an equivalence relation $(U, \sim)$ is the problem to decide, for any given $u, v \in U$, whether $u \sim v$ holds. Later we will show that the Church-Rosser property is equivalent to confluence.

We shall write $x \downarrow y$ in case there is a z such that $x \xrightarrow{*} z \xleftarrow{*} y$.

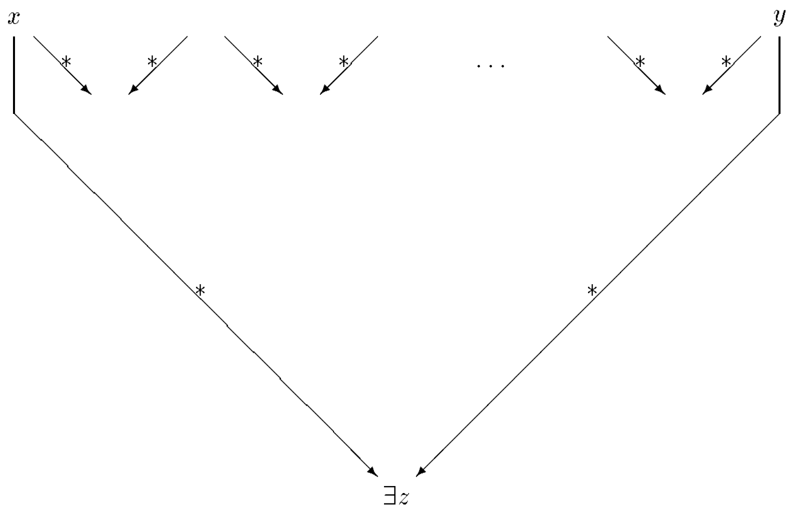
Definition 2.3.1. (Confluence, local confluence, Church-Rosser property). Let $(U, \longrightarrow)$ be a reduction system.

1. $\longrightarrow$ is called *confluent*, if $u \xrightarrow{*} x$ and $u \xrightarrow{*} y$ implies $x \downarrow y$.
2. $\longrightarrow$ is called *locally confluent*, if $u \longrightarrow x$ and $u \longrightarrow y$ implies $x \downarrow y$.
3. $\longrightarrow$ has the *Church-Rosser property*, if $x \xleftrightarrow{*} y$ implies $x \downarrow y$.

So confluence und local confluence mean

$$\begin{tikzcd}
& u \arrow[dl, "*"'] \arrow[dr, "*"] & \\
x \arrow[dr, "*"'] & & y \arrow[dl, "*"] \\
& \exists z &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& u \arrow[dl] \arrow[dr] & \\
x \arrow[dr, "*"'] & & y \arrow[dl, "*"] \\
& \exists z &
\end{tikzcd}$$

and the Church-Rosser property might be visualized by



A situation $x \overset{*}{\longleftarrow} u \overset{*}{\longrightarrow} y$ is called *divergence*, and $x \longleftarrow u \longrightarrow y$ is called *local divergence* or *peak*.

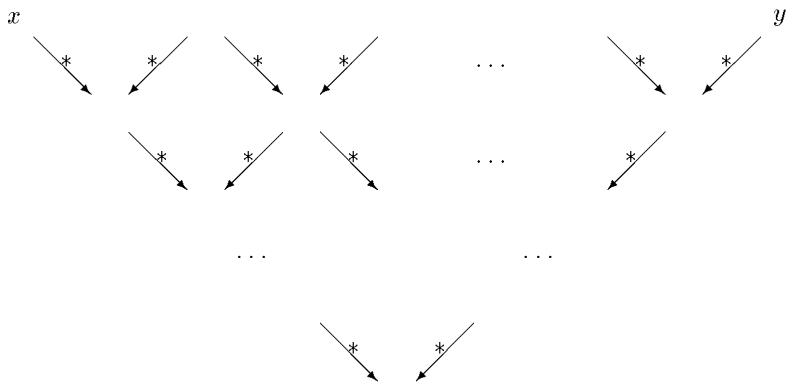
Clearly confluence implies local confluence. However, the reverse implication does not hold. To see this, consider the reduction system given by

$$a \longleftarrow b \longrightarrow c \quad \text{and} \quad b \longleftarrow c \longrightarrow d.$$

Then we have just these two local divergences. Both can be joined: $a \downarrow c$ follows from $a \longleftarrow b \longleftarrow c$, and $b \downarrow d$ follows from $b \longrightarrow c \longrightarrow d$. But the system is not confluent: consider the divergence $a \longleftarrow b \overset{*}{\longrightarrow} d$.

Note that we have a non-terminating reduction system here. This is necessary, since we will show below in lemma 2.3.6 that for any terminating reduction systems the properties of local confluence and of confluence coincide.

We now show that confluence and the Church–Rosser property are equivalent. This can be seen easily from



We now give a more formal proof, which also shows the equivalence of the Church-Rosser property to another useful property, the *one-sided local confluence*.

Definition 2.3.2. Let $(U, \longrightarrow)$ be a reduction system. $\longrightarrow$ is called *one-sided locally confluent* if $u \longrightarrow x$ and $u \overset{*}{\longrightarrow} y$ implies $x \downarrow y$.

So one-sided local confluence means

$$\begin{tikzcd}
& u \arrow[dl] \arrow[dr, "*"] & \\
x \arrow[dr, "*"] & & y \arrow[dl, "*"'] \\
& \exists z &
\end{tikzcd}$$

Theorem 2.3.3. *Let $(U, \longrightarrow)$ be a reduction system. Then the following are equivalent.*

1. *$\longrightarrow$ is confluent.*
2. *$\longrightarrow$ is one-sided locally confluent.*
3. *$\longrightarrow$ has the Church-Rosser property.*

Proof. (1) $\Longrightarrow$ (2) is obvious.

(2) $\Longrightarrow$ (3). We show $x \overset{n}{\longleftrightarrow} y \Longrightarrow x \downarrow y$ by induction an n. *Case* 0. Obvious. *Case* $n+1$. In case $x \overset{n}{\longleftrightarrow} y \longrightarrow z$ and $x \overset{n}{\longleftrightarrow} y \longleftarrow z$ consider the following two figures.

$$\begin{tikzcd}
x \arrow[rr, leftrightarrow, "n"] \arrow[dr, "*"'] & \arrow[d, phantom, "\mathrm{IH}"] & y \arrow[dl, "*"] \arrow[dr] & \\
& u \arrow[dr, "*"'] \arrow[rr, phantom, "\mathrm{Ann.}"] & & z \arrow[dl, "*"] \\
& & \exists v &
\end{tikzcd}
\qquad
\begin{tikzcd}
x \arrow[rr, leftrightarrow, "n"] \arrow[dr, "*"'] & \arrow[d, phantom, "\mathrm{IH}"] & y \arrow[dl, "*"] & z \arrow[l] \\
& u & &
\end{tikzcd}$$

(3) $\Longrightarrow$ (1). Let $x \overset{*}{\longleftarrow} u \overset{*}{\longrightarrow} y$. Then by definition $x \overset{*}{\longleftrightarrow} y$, hence $x \downarrow y$ by the Church-Rosser property. □

The next lemma relates the properties of termination and confluence to the existence of unique normal forms.

Lemma 2.3.4. *Let $(U, \longrightarrow)$ be a reduction system.*

1. *If $\longrightarrow$ is terminating and confluent, then every $u \in U$ has exactly one normal form.*
2. *If every $u \in U$ has exactly one normal form, then $\longrightarrow$ is confluent, but not necessarily terminating.*

Proof. 1. Existence of a normal form follows immediately from termination. Uniqueness can be seen as follows. Assume that x, y are both normal forms of u. Since $x \downarrow y$ by confluence, and since x, y are both irreducible, we must have $x = y$.

2. To prove confluence, assume $x \overset{*}{\longleftarrow} u \overset{*}{\longrightarrow} y$. Let $x\downarrow$, $y\downarrow$, $u\downarrow$ be the unique normal forms of x, y, u. Then $x\downarrow = u\downarrow = y\downarrow =: z$, hence $x = y = z$:

$$\begin{tikzcd}
& u \arrow[dl, "*"'] \arrow[dr, "*"] & \\
x \arrow[dr, "*"] & & y \arrow[dl, "*"'] \\
& z &
\end{tikzcd}$$

In the reduction system below every element has exactly one normal form (namely c), but it is not terminating.

$$\begin{tikzcd}
a \arrow[rr, shift left] \arrow[dr] & & b \arrow[ll, shift left] \arrow[dl] \\
& c &
\end{tikzcd}$$

□

Definition 2.3.5. A terminating and confluent reduction system is called *convergent*.

As already noted, it is often much easier to prove local confluence than confluence. We now show that for terminating reduction systems both notions coincide.

Lemma 2.3.6 (Newman [23]). *Let $(U, \longrightarrow)$ be a terminating reduction system. If $\longrightarrow$ is locally confluent, then it is also confluent.*

Proof. Call $u \in \mathcal{E}$ *good* if it satisfies the confluence property, i.e. if whenever $x \xleftarrow{*} u \xrightarrow{*} y$, then $x \downarrow y$. We show that every $u \in \mathcal{E}$ is good, by transfinite induction on the well-founded partial order $\longrightarrow^+$. So let u be given and assume

$$\forall v[u \longrightarrow^+ v \implies v \text{ is good}].$$

We must show that u is good, so assume $x \xleftarrow{*} u \xrightarrow{*} y$. We may further assume that there are v_1, v_2 such that $x \xleftarrow{*} v_1 \longleftarrow u \longrightarrow v_2 \xrightarrow{*} y$, for otherwise the claim is trivial. But then the claim follows from the assumed local confluence and the IH for v_1 and v_2, as shown in the picture below.

$$\begin{tikzcd}
 & & u \arrow[dl] \arrow[dr] & & \\
 & v_1 \arrow[dl, "*"'] \arrow[dr, "*"] & \text{loc. conf.} & v_2 \arrow[dl, "*"'] \arrow[dr, "*"] & \\
x \arrow[dr, "*"'] & \mathrm{IH}(v_1) & \exists z_1 \arrow[dl, "*"] & & y \arrow[ddll, "*"] \\
 & \exists z_2 \arrow[dr, "*"'] & & \mathrm{IH}(v_2) & \\
 & & \exists z & &
\end{tikzcd}$$

□

As an application of Newman's Lemma we prove confluence of $\longrightarrow_{\beta\eta}$ and $\longrightarrow_\beta$. It suffices to show

Lemma 2.3.7. *$\longrightarrow_{\beta\eta}$ and $\longrightarrow_\beta$ are locally confluent.*

Proof. Assume $N_0 \longleftarrow M \longrightarrow N_1$. We show $N_0 \downarrow N_1$ by induction on M. If there are two inner reductions both on the same subterm, then the claim follows from the induction hypothesis using lemma 2.2.2(1) and (2). If they are on distinct subterms, then the subterms do not overlap and the claim is obvious. It remains to deal with the case of a head conversion together with an inner conversion. For $\mapsto_\beta$ we have

$$\begin{tikzcd}
 & (\lambda x M)N\boldsymbol{L} \arrow[dl] \arrow[dr] & \\
M[x:=N]\boldsymbol{L} \arrow[dr] & & (\lambda x M')N\boldsymbol{L} \arrow[dl] \\
 & M'[x:=N]\boldsymbol{L} &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & (\lambda x M)N\boldsymbol{L} \arrow[dl] \arrow[dr] & \\
M[x:=N]\boldsymbol{L} \arrow[dr, "*"] & & (\lambda x M)N'\boldsymbol{L} \arrow[dl] \\
 & M[x:=N']\boldsymbol{L} &
\end{tikzcd}$$

$$\begin{tikzcd}
 & (\lambda x M)N\boldsymbol{L} \arrow[dl] \arrow[dr] & \\
M[x:=N]\boldsymbol{L} \arrow[dr] & & (\lambda x M')N\boldsymbol{L}' \arrow[dl] \\
 & M'[x:=N]\boldsymbol{L}' &
\end{tikzcd}$$

where in the left picture we have used lemma 2.2.2. For $\mapsto_\eta$ we obtain because of the neutrality of M

$$\begin{tikzcd}
& \lambda x.Mx \arrow[dl] \arrow[dr] & \\
M \arrow[dr] & & \lambda x.M'x \arrow[dl] \\
& M' &
\end{tikzcd}$$

where in the lower right arrow we have used an η-conversion or in case $M' = \lambda x N$ an inner β-conversion. □

2.4 Extension to Full First Order Logic

All we have done up to now can easily be extended to full first order logic. A term is again called *neutral* if it is neither an abstraction nor a pair. Terms are characterized by

$$M ::= u\boldsymbol{M} \mid \langle M, N\rangle \mid \langle M, N\rangle i\boldsymbol{L} \mid \lambda u M \mid (\lambda u M)N\boldsymbol{L} \mid \lambda x M \mid (\lambda x M)t\boldsymbol{N},$$

where we now allow that in $\boldsymbol{M}$ und $\boldsymbol{L}$ there are occurrences of $0, 1$. In case $\lambda x M$ the variable condition needs to be satisfied, i.e. $x \notin \mathsf{FV}(A)$ for all $u^A \in \mathsf{FA}(M)$. The rules of β- and η-conversion are now

$$\begin{aligned}
\langle M_0, M_1\rangle i\boldsymbol{N} &\mapsto_\beta M_i\boldsymbol{N}, \\
(\lambda u M)N\boldsymbol{L} &\mapsto_\beta M[u := N]\boldsymbol{L}, \\
(\lambda x M)t\boldsymbol{N} &\mapsto_\beta M[x := t]\boldsymbol{N}, \\
\langle M0, M1\rangle &\mapsto_\eta M \quad \text{if } M \text{ is neutral}, \\
\lambda u.Mu &\mapsto_\eta M \quad \text{if } u \notin \mathsf{FA}(M) \text{ and } M \text{ is neutral}, \\
\lambda x.Mx &\mapsto_\eta M \quad \text{if } M \text{ is neutral}.
\end{aligned}$$

The term closure $\longrightarrow_\gamma$ of a conversion rule $\mapsto_\gamma$ and hence the reduction relations $\longrightarrow_\beta$, $\longrightarrow_\eta$ and $\longrightarrow_{\beta\eta} := \longrightarrow_\beta \cup \longrightarrow_\eta$ are defined as before. The properties of substitution, i.e. lemma 2.2.2, remain valid. γ-*normal* terms for $\gamma \in \{\beta, \eta, \beta\eta\}$ are defined as before. β-normal terms are now characterized by

$$M ::= u\boldsymbol{M} \mid \langle M, N\rangle \mid \lambda u M \mid \lambda x M,$$

where in case $\lambda x M$ the variable condition needs to be observed. For $\beta\eta$-normal terms we need in case $\langle M, N\rangle$ the additional restriction that we should not have both $M = L0$ and $N = L1$. The proof of termination of $\longrightarrow_{\beta\eta}$ can easily be extended; also the additional cases in the proof of local confluence of $\longrightarrow_{\beta\eta}$ and $\longrightarrow_\beta$ are very easy.

2.5 Applications

We now want to draw some conclusions from the fact that every derivation can be brought into β-normal form. For this purpose we have to analyse more closely the form of normal derivations.

With $k, l \in \{0, 1\}^*$ we denote *nodes* in a term M. $\mathrm{Node}(M)$ is the set of all nodes in M, and ε denotes the empty list, i.e. the root node. $k0$ and $k1$ are the extensions of the node k by 0 or 1. The subterm of M at node k is denoted by M/k, and $M[N]_k$ results from M by replacing the subterm at node k by N. We write $k \leq l$ if k is a prefix of l. k and l are called *independent* (written $k \parallel l$) if $k \not\leq l$ and $l \not\leq k$. Furthermore let

$$\begin{aligned}
\mathrm{Leafnode}(M) &:= \{\, k \in \mathrm{Node}(M) \mid M/k \text{ variable or constant} \,\} \\
\mathrm{Elimnode}(M) &:= \{\, k \in \mathrm{Node}(M) \mid M/k \text{ formed by an elimination rule} \,\} \\
\mathrm{Intronode}(M) &:= \{\, k \in \mathrm{Node}(M) \mid M/k \text{ formed by an introduction rule.} \,\}
\end{aligned}$$

For any $k \in \mathrm{Elimnode}(M) \cup \mathrm{Leafnode}(M)$ we define its associated minimal node by

$$\mathrm{minnode}_M(\varepsilon) := \varepsilon.$$
$$\mathrm{minnode}_M(ki) := \begin{cases} \mathrm{minnode}_M(k), & \text{if } i = 0 \text{ and } k \in \mathrm{Elimnode}(M); \\ ki, & \text{if } i = 1 \text{ and } k \in \mathrm{Elimnode}(M), \text{ or} \\ & k \in \mathrm{Intronode}(M). \end{cases}$$

Clearly $\mathrm{minnode}_M(k) \in \mathrm{Elimnode}(M) \cup \mathrm{Leafnode}(M)$. Let

$$\mathrm{Minnode}(M) := \{\, k \in \mathrm{Elimnode}(M) \cup \mathrm{Leafnode}(M) \mid \mathrm{minnode}_M(k) = k \,\}.$$

For any $k \in \mathrm{Node}(M)$ we define its associated end node by

$$\mathrm{endnode}_M(\varepsilon) := \varepsilon.$$
$$\mathrm{endnode}_M(ki) := \begin{cases} \mathrm{endnode}_M(k), & \text{if } k \in \mathrm{Intronode}(M), \text{ or} \\ & i = 0 \text{ and } k \in \mathrm{Elimnode}(M); \\ ki, & \text{if } i = 1 \text{ and } k \in \mathrm{Elimnode}(M). \end{cases}$$

Let

$$\mathrm{Endnode}(M) := \{\, k \in \mathrm{Node}(M) \mid \mathrm{endnode}_M(k) = k \,\}.$$

For β-normal terms M the leaf nodes and the minimal nodes are in a bijective correspondence. Note however that two minimal nodes can yield the same end node.

Let M be a term and $k \in \mathrm{Leafnode}(M)$. Then

$$\{\, l \in \mathrm{Node}(M) \mid k \geq l \geq \mathrm{endnode}_M(k) \,\}$$

is called the *branch* in M determined by the leaf node k. In particular, $\mathrm{minnode}_M(k)$ is an element of the branch determined by k. In case $\mathrm{endnode}_M(k) = \varepsilon$ the branch is called a *main branch*. In a β-normal term any branch $k_1 \geq \dots \geq k_n$ has a particularly perspicious form: all elimination rules must come before all introduction rules.

In the case of a normal derivation term we want to draw some conclusions on the formulas attached to the nodes. To do this we need the following notions. The formulas A, B are called *immediate subformulas* of $A \wedge B$ and $A \to B$, and for every term t the formula $A[x := t]$ is an immediate subformula of $\forall x A$. The relation "A is a *subformula* of B" is the reflexive transitive closure of the immediate subformula relation.

Lemma 2.5.1. *If $M^B[u_1^{A_1}, \dots, u_n^{A_n}]$ is a normal derivation not ending with an introduction, then B is a subformula of some formula A_i.*

Proof. Induction on M. *Case* $M^B = M_0^{D \to B} M_1^D$. Since M is normal, M_0 cannot end with an introduction. So by induction hypothesis $D \to B$ and hence B are subformulas of some A_i. The other cases (i.e. $M^B = N^{B \wedge D} 0$, $M^B = N^{D \wedge B} 1$, $M^B = N^{\forall x D} t$ with $B = D[x := t]$ and $M = u_i^{A_i}$ with $B = A_i$) are treated similarly. □

Theorem 2.5.2 (Subformula property). *If $M^B[u_1^{A_1}, \dots, u_n^{A_n}]$ is a normal derivation and N^C is a subdervation of M^B, then C is a subformula of B or of some A_i.*

Proof. Induction on M. We may assume $N^C \neq M^B$. *Case* $M^B = M_0^{D \to B} M_1^D$. By induction hypothesis C is subformula of $D \to B$ or of some A_i. Since M is normal, M_0 cannot end with an introduction. So by the previous lemma $D \to B$ is a subformula of some A_i. Hence also C is a subformula of some A_i. The cases $M^B = N^{B \wedge D} 0$, $M^B = N^{D \wedge B} 1$ and $M^B = N^{\forall x D} t$ with $B = D[x := t]$ are treated similarly.

Case $M^B = \lambda u^{B_0} M_1^{B_1}$ with $B = B_0 \to B_1$. By induction hypothesis C is a subformula of B_1, B_0 or some A_i. Hence C is a subformula of $B_0 \to B_1$ or some A_i. The cases $M = \lambda x M_0$ and $M = \langle M_0, M_1 \rangle$ are treated similarly. □

Lemma 2.5.3. *If $M[u_1^{\forall \boldsymbol{x}_1 A_1}, \dots, u_n^{\forall \boldsymbol{x}_n A_n}] \colon \forall \boldsymbol{y} \forall y B_0$ is a normal derivation with quantifier-free formulas $A_1, \dots, A_n, B_0$ not ending with an introduction, then M is of the form $u_i \boldsymbol{t}$.*

Proof. Induction on M. Let $B := \forall \boldsymbol{y} \forall y B_0$. In *case* $M^B = u_i$ the claim is obvious. *Case* $M^B = M_0^{D \to B} M_1^D$. Since M is normal, M_0 cannot end with an introduction. Hence by lemma 2.5.1 $D \to B$ is a subformula of a $\forall \boldsymbol{x}_i A_i$. But this is impossible, since B begins with $\forall$; hence this case cannot occur. The cases $M^B = N^{B \wedge D} 0$ and $M^B = N^{D \wedge B} 1$ are treated similarly. *Case* $M^B = N^{\forall z D} t$ with $B = D[z := t]$. Then we have $\forall z D = \forall z \forall \boldsymbol{y} \forall y D_0$ with quantifier-free D_0. So by induction hypothesis $N = u_i \boldsymbol{t}$, hence $M = u_i \boldsymbol{t} t$. □

Theorem 2.5.4 (Herbrand). *Let $M[u_1^{\forall \boldsymbol{x}_1 A_1}, \ldots, u_n^{\forall \boldsymbol{x}_m A_m}] \colon B$ be a normal derivation with quantifier-free formulas $A_1, \ldots, A_m, B$. Then M^B does not contain an all-introduction, and every occurrence of an assumption variable u_i in M is in a context*

$$\frac{u_i \colon \forall \boldsymbol{x}_i A_i \qquad \boldsymbol{t}_{ij}}{A_i[\boldsymbol{x}_i := \boldsymbol{t}_{ij}]}.$$

In particular one finds terms $\boldsymbol{t}_{11}, \ldots, \boldsymbol{t}_{1n_1}, \ldots, \boldsymbol{t}_{m1}, \ldots, \boldsymbol{t}_{mn_m}$ such that

$$\begin{aligned} &A_1[\boldsymbol{x}_1 := \boldsymbol{t}_{11}], \ldots, A_1[\boldsymbol{x}_1 := \boldsymbol{t}_{1n_1}], \ldots, \\ &A_m[\boldsymbol{x}_m := \boldsymbol{t}_{m1}], \ldots, A_m[\boldsymbol{x}_m := \boldsymbol{t}_{mn_m}] \vdash B. \end{aligned}$$

Proof. Part 1. In *case* $M = u_i$ there is nothing to show. *Case* $M^B = M_0^{D \to B} M_1^D$. Since M is normal, M_0 cannot end with an introduction. By lemma 2.5.1 $D \to B$ is subformula of a $\forall \boldsymbol{x}_i A_i$. Hence $D \to B$ is quantifier-free, and the induction hypothesis can be applied. The cases $M^B = N^{B \wedge D} 0$ and $M^B = N^{D \wedge B} 1$ are treated similarly. *Case* $M^B = N^{\forall z D} t$ with $B = D[z := t]$, hence D quantifier-free. Since M^B is normal, N cannot end with an introduction. By lemma 2.5.3 N then must have the form $u_i \boldsymbol{t}$, hence $M = u_i \boldsymbol{t} t$.

Case $M^B = \lambda u^{B_0} M_1^{B_1}$ with $B = B_0 \to B_1$. Then B_0 and B_1 are quantifier-free, and the induction hypothesis yields the claim. The case $M = \langle M_0, M_1 \rangle$ is treated similarly.

Part 2. If one replaces all occurrences of $u_i \boldsymbol{t}_{ij}$ by new assumption variables $v_i \colon A_i[\boldsymbol{x}_i := \boldsymbol{t}_{ij}]$, one obtains a correct derivation, since by part (1) M does not contain an all-introduction and hence none of the variables in $\boldsymbol{t}_{ij}$ is bound further down. □

2.6 Normal Versus Non-Normal Derivations

We now show that the requirement to give a normal derivation of a derivable formula can sometimes be unrealistic. Following Statman [27] and Orevkov [24] we give examples of formulas C_k which are easily derivable with non-normal derivations (whose number of nodes is linear in k), but which require a non-elementary (in k) number of nodes in any normal derivation.

The example is related to Gentzen's proof in [16] of transfinite induction up to ω_k in arithmetic. There the function $y \oplus \omega^x$ plays a crucial role, and also the assignment of a "lifting"-formula A^+ to any formula A, by

$$A^+ := \forall y. (\forall z {\prec} y \, A[x := z]) \to \forall z {\prec} y {\oplus} \omega^x \; A[x := z].$$

Here we consider the numerical function $y + 2^x$ instead, and axiomatize its graph by means of Horn clauses. The formula C_k expresses that from these axioms the existence of 2_k follows (where $2_0 := 1$ and $2_{k+1} := 2^{2_k}$). A short, non-normal proof of this fact can then be given by a modification of Gentzen's idea, and it is easily seen that any normal proof of C_k must contain at least 2_k nodes.

The derivations to be given make heavy use of the existential quantifier $\exists$ defined by $\neg \forall \neg$ (cf. section 1.5).

Note that the stability assumption $\neg\neg B \to B$ is not needed if B does not contain an atom $\neq \bot$ as a strictly positive subformula[1]. This will be the case for the derivations below, where B will always be an existential formula.

Let us now fix our language. We use a ternary relation symbol R to represent the graph of the function $y + 2^x$; so $R(y, x, z)$ is intended to mean $y + 2^x = z$. We now axiomatize R by means of Horn clauses. For

[1] The formulas A, B are *immediate strictly positive subformulas* of $A \wedge B$, B is an immediate strictly positive subformula of $A \to B$, and for every term t the formula $A[x := t]$ is an immediate strictly positive subformula of $\forall x A$. The relation "A is a *strictly positive subformula* of B" is the reflexive transitive closure of this relation

simplicity we use a unary function symbol s (to be viewed as the successor function) and a constant 0; one could use logic without function symbols instead – as Orevkov does –, but this makes the formulas somewhat less readable and the proofs less perspicious. Note that Orevkov's result is an adaption of a result of Statman [27] for languages containing function symbols.

$$\mathsf{Hyp}_1 \colon \forall y R(y, 0, s(y))$$
$$\mathsf{Hyp}_2 \colon \forall y, x, z, z_1 . R(y, x, z) \to R(z, x, z_1) \to R(y, s(x), z_1)$$

The goal formula then is

$$C_k := \exists z_k, \dots, z_0 . R(0, 0, z_k) \wedge R(0, z_k, z_{k-1}) \wedge \dots \wedge R(0, z_1, z_0).$$

To obtain the short proof of the goal formula C_k we use formulas A_i with a free parameter x; for ease in reading we write $A[r]$ instead of $A[x := r]$.

$$A_0 := \forall y \exists z\, R(y, x, z),$$
$$A_{i+1} := \forall y . A_i[y] \to \exists z . A_i[z] \wedge R(y, x, z).$$

Lemma 2.6.1. $\vdash \mathsf{Hyp}_1 \to \mathsf{Hyp}_2 \to A_i[0]$.

Proof. We give an informal argument, which can easily be converted into a formal proof. Note that the existence elimination lemma is used only with existential formulas as conclusions. Hence it is not necessary to use stability axioms and we have a derivation in minimal logic.

Case $i = 0$. Obvious by Hyp_1.

Case $i = 1$. Let x with $A_0[x]$ be given. It is sufficient to show $A_0[s(x)]$, that is $\forall y \exists z_1 R(y, s(x), z_1)$. So let y be given. We know

$$A_0[x] = \forall y \exists z\, R(y, x, z). \tag{2.1}$$

Applying (2.1) to our y gives z such that $R(y, x, z)$. Applying (2.1) again to this z gives z_1 such that $R(z, x, z_1)$. By Hyp_2 we obtain $R(y, s(x), z_1)$.

Case $i + 2$. Let x with $A_{i+1}[x]$ be given. It suffices to show $A_{i+1}[s(x)]$, that is $\forall y . A_i[y] \to \exists z . A_i[z] \wedge R(y, s(x), z)$. So let y with $A_i[y]$ be given. We know

$$A_{i+1}[x] = \forall y . A_i[y] \to \exists z_1 . A_i[z_1] \wedge R(y, x, z_1). \tag{2.2}$$

Applying (2.2) to our y gives z such that $A_i[z]$ and $R(y, x, z)$. Applying (2.2) again to this z gives z_1 such that $A_i[z_1]$ and $R(z, x, z_1)$. By Hyp_2 we obtain $R(y, s(x), z_1)$. □

Note that the derivations given have a fixed length, independent of i.

Lemma 2.6.2. $\vdash \mathsf{Hyp}_1 \to \mathsf{Hyp}_2 \to C_k$.

Proof. We give an informal argument, which can easily be converted into a formal proof. Again the existence elimination lemma is used only with existential formulas as conclusions, and hence we have a derivation in minimal logic.

$A_k[0]$ applied to 0 and $A_{k-1}[0]$ yields z_k with $A_{k-1}[z_k]$ and $R(0, 0, z_k)$.

$A_{k-1}[z_k]$ applied to 0 and $A_{k-2}[0]$ yields z_{k-1} with $A_{k-2}[z_{k-1}]$ and $R(0, z_k, z_{k-1})$.

$A_1[z_2]$ applied to 0 and $A_0[0]$ yields z_1 with $A_0[z_1]$ and $R(0, z_2, z_1)$.

$A_0[z_1]$ applied to 0 yields z_0 with $R(0, z_1, z_0)$. □

Note that the derivations given have length linear in k.

We want to compare the length of this derivation of C_k with the length of an arbitrary normal derivation.

Lemma 2.6.3. *Any normal derivation of C_k from Hyp_1 and Hyp_2 has at least 2_k nodes.*

Proof. Let a normal derivation M of falsity $\bot$ from Hyp_1, Hyp_2 and the additional hypothesis

$$u \colon \forall z_k, \dots, z_0 . R(0,0,z_k) \to R(0,z_k,z_{k-1}) \to \dots \to R(0,z_1,z_0) \to \bot$$

be given. We may assume that M does not contain free object variables (otherwise substitute them by 0). The main branch of M must begin with u, and its side premises are all of the form $R(0, s^n(0), s^k(0))$.

Observe that any normal derivation of $R(s^m(0), s^n(0), s^k(0))$ from Hyp_1, Hyp_2 and u has at least 2^n occurrences of Hyp_1 and is such that $k = m + 2^n$. This can be seen easily by induction on n. Note also that such a derivation cannot involve u.

If we apply this observation to the above derivations of the side premises we see that they derive

$$R(0,0,s^{2^0}(0)), \quad R(0,s^{2^0}(0),s^{2^{2^0}}(0)), \quad \dots \quad R(0,s^{2_{k-1}}(0),s^{2_k}(0)).$$

The last of these derivations uses at least $2^{2_{k-1}} = 2_k$-times Hyp_1. □

2.7 Notes

The proof of strong normalization for typed λ terms is due to Ralph Matthes and Felix Joachimski.

2.8 Problems

Problem 2.8.1. Give derivation terms for

$$\vdash A \to \exists x A,$$
$$\vdash (\neg\neg B \to B) \to \exists x A \to (\forall x . A \to B) \to B \quad \text{if } x \notin \mathsf{FV}(B).$$

Solution 2.8.2.

$$\lambda u^A \lambda v^{\forall x \neg A} . v x u,$$
$$\lambda u^{\neg\neg B \to B} \lambda v^{\neg \forall x \neg A} \lambda w^{\forall x . A \to B} . u \lambda u_2^{\neg B} . v \lambda x \lambda u_1^A . u_2 (w x u_1).$$

Problem 2.8.3. For a fixed ground ι define the *iterated types* by $0\iota := \iota$, $(k+1)\iota := k\iota \to k\iota$, and the *Church numerals* of type $k\iota$ by

$$I_n^{k\iota} := \lambda y^{k\iota \to k\iota} \lambda x^{k\iota} . \, y^n(x)$$

where $y^n(x)$ stands for $y(y(\dots(yx)\dots))$ with n occurrences of y. Write I_n^k for $I_n^{k\iota}$. Using the abbreviations $M \circ N := \lambda x . M(Nx)$ and $M =_\beta N \; := \; M \overset{*}{\longleftrightarrow}_\beta N$ show that

a. $(I_n^k y) \circ (I_m^k y) =_\beta I_{n+m}^k y$,
b. $I_n^k \circ I_m^k =_\beta I_{nm}^k$,
c. $I_n^{k+1} I_m^k y =_\beta I_{n^m}^k$ for $m > 0$.

3. The Computational Content of Proofs

It is well known that proofs may have computational content. In the present chapter we make this more precise and define a general operator from derivations to λ-terms with the intention that the λ-term assigned should represent the (functional) program implicit in the proof. The method we employ here is the so-called modified realizability interpretation.

3.1 Gödel's System T

We introduce Gödel's system T of primitive recursive functionals of finite type, formulated as a simply typed lambda calculus with higher type recursion constants.

Types are as for the simply typed lambda calculus, but with concrete ground types ι for the natural numbers and o for the boolean objects true and false.

$$\iota \mid o \mid \rho \times \sigma \mid \rho \to \sigma.$$

The *constants* are

$$\mathsf{true}^o \mid \mathsf{false}^o \mid 0^\iota \mid \mathrm{S}^{\iota\to\iota} \mid \mathsf{Rec}_{o,\rho} \mid \mathsf{Rec}_{\iota,\rho}.$$

$\mathsf{Rec}_{\iota,\rho}$ is the *primitive recursion operator* of type $\rho \to (\iota \to \rho \to \rho) \to \iota \to \rho$. $\mathsf{Rec}_{o,\rho}$ is of type $\rho \to \rho \to o \to \rho$ and represents *boolean induction*, i.e. definition by cases. Instead of $\mathsf{Rec}_{o,\rho}MNK$ we will often write **if** K **then** M **else** N.

Terms are

$$x^\rho \mid c^\rho \ (c^\rho \text{ a constant}) \mid \langle M, N\rangle \mid \pi_0(M) \mid \pi_1(M) \mid \lambda x^\rho M \mid MN$$

with the usual typing rules. The *conversions* are those for the simply typed lambda calculus, plus some new ones for the recursion operators. We write $K+1$ for $\mathrm{S}K$.

$$\begin{array}{rcl}
\mathsf{Rec}_{o,\rho}MN\,\mathsf{true} & \mapsto_{\mathsf{Rec}} & M \\
\mathsf{Rec}_{o,\rho}MN\,\mathsf{false} & \mapsto_{\mathsf{Rec}} & N \\
\mathsf{Rec}_{\iota,\rho}MN0 & \mapsto_{\mathsf{Rec}} & M \\
\mathsf{Rec}_{\iota,\rho}MN(K+1) & \mapsto_{\mathsf{Rec}} & NK(\mathsf{Rec}_{\iota,\rho}MNK)
\end{array}$$

We will prove that also for this system of terms every term strongly normalizes. Since again the normal form is uniquely determined, the relation $M =_{\beta\mathsf{Rec}} N$ is decidable (by normalizing M and N). By identifying $=_{\beta\mathsf{Rec}}$-equal terms (i.e. treating equations on the meta level) we can greatly simplify many formal derivations.

Tait's Proof of Termination for the Simply Typed Lambda Calculus

We first give yet another proof of *termination* of the simply typed lambda calculus, using method due to W.W. Tait. This proof will later be extended to Gödel's system T.

Tait's proof rests on defining so-called *strong computability predicates*. We present the proof here in a form which avoids intuitive arguments concerning reduction sequences and therefore is suitable for formalization in a theory of inductive definitions.

We begin with a definition of strongly normalizable terms, by a strictly positive induction.

Definition 3.1.1. If all M' such that $M \longrightarrow_\beta M'$ are strongly normalizable (sn), then so is M.

Obviously M is sn if and only if every reduction sequence starting with M terminates after a finite number of steps. This can be seen as follows. $\Longrightarrow$. Induction on the definition of "strongly normalizable". Consider a reduction sequence starting with M and therein the first reduct M'. The IH for M' yields the assertion. $\Longleftarrow$. By induction on the length of the longest reduction sequence (König's lemma).

We note a number of the properties of the notion "strongly normalizable", to be used below.

$$\text{If all terms } \boldsymbol{M} \text{ are sn, then so is } x\boldsymbol{M}. \tag{3.1}$$

Proof. Induction on the definition of sn for $\boldsymbol{M}$. Let $x\boldsymbol{M} \longrightarrow_\beta N$ be given. It suffices to show that N is sn. From $x\boldsymbol{M} \longrightarrow_\beta N$ it follows that $N = x\boldsymbol{M}'$, where $\boldsymbol{M}'$ arises by substitution of M_i by M_i' with $M_i \longrightarrow_\beta M_i'$. It is to be proved that $x\boldsymbol{M}'$ is sn. This follows from the IH for $\boldsymbol{M}'$. □

$$\text{If } Mx \text{ is sn, then so is } M. \tag{3.2}$$

Proof. Induction on the definition of sn for Mx. Let $M \longrightarrow_\beta M'$ be given. It suffices to show that M' is sn. From $M \longrightarrow_\beta M'$ we get $Mx \longrightarrow_\beta M'x$. The IH for $M'x$ then yields that M' is sn. □

We now define when a term M^ρ is strongly computable (sc), by induction on the type ρ.

Definition 3.1.2. – M^ι is sc if M^ι is sn.
– $M^{\rho\to\sigma}$ is sc if for every sc N^ρ also $(MN)^\sigma$ is sc.

A term M is called *strongly computable under substitution* if for any sc terms $\boldsymbol{N}$ also $M[\boldsymbol{x} := \boldsymbol{N}]$ is sc.

We note a property of the notion "strongly computable" which will be used below.

$$M \text{ is sc if and only if } M\boldsymbol{N} \text{ is sc for all sc } \boldsymbol{N}. \tag{3.3}$$

Proof. Induction on the length of $\boldsymbol{N}$. □

Lemma 3.1.3. *a. Every sc term M^ρ is sn.*
b. If $\boldsymbol{M}$ are sn, then $(x\boldsymbol{M})^\rho$ is sc.

Proof. By simultaneous induction on the type ρ. *Case* ι. a. By definition. b. By (3.1).

Case $\rho \to \sigma$ a. Let $M^{\rho\to\sigma}$ be sc. By IHb (with $\boldsymbol{M}$ empty) and the definition of strong computability $(Mx)^\sigma$ is sc. By IHa Mx then is sn. By (3.2) M is sn too. b. Consider $(x\boldsymbol{M})^{\rho\to\sigma}$ with $\boldsymbol{M}$ sn. Let N^ρ be sc. We have to show that $(x\boldsymbol{M}N)^\sigma$ is sc. By IHa N is sn, hence $x\boldsymbol{M}N$ is sc by IHb. □

If $M \longrightarrow_\beta M'$ and M is sn, then by the definition of strong normalizability also M' is sn. We now show that the corresponding assertion is also valid for strong computability.

Lemma 3.1.4. *If $M \longrightarrow_\beta M'$ and M is sc, then so is M'.*

Proof. We use (3.3). Let $\boldsymbol{N}$ be a list of sc terms such that $M\boldsymbol{N}$ is of ground type. Then $M\boldsymbol{N}$ is sc by (3.3), hence also sn. Furthermore we have $M\boldsymbol{N} \longrightarrow_\beta M'\boldsymbol{N}$. By definition of strong normalizablility $M'\boldsymbol{N}$ is sn. Thus by (3.3) M' is sc. □

Lemma 3.1.5. *Let N be sn. If $M[x := N]\boldsymbol{L}$ is sn and of a ground type, then so is $(\lambda x\, M)N\boldsymbol{L}$.*

Proof. By induction on the definition of strong normalizability for N and $M[x := N]\boldsymbol{L}$. So we consider all the reducts K of $(\lambda x\, M)N\boldsymbol{L}$. Clearly it suffices to show that every such K is sn.

Case $K = M[x := N]\boldsymbol{L}$, i.e. we have an outer β-conversion. Hence K is sn by assumption.

Case $K = (\lambda x\, M')N\boldsymbol{L}$ with $M \longrightarrow_\beta M'$. Then we have $M[x := N]\boldsymbol{L} \longrightarrow_\beta M'[x := N]\boldsymbol{L}$. By definition of strong normalizability $M'[x := N]\boldsymbol{L}$ is sn. Hence by IH K is sn.

Case $K = (\lambda x\, M)N'\boldsymbol{L}$ with $N \longrightarrow_\beta N'$. Then we have $M[x := N]\boldsymbol{L} \longrightarrow_\beta^* M[x := N']\boldsymbol{L}$. By definition of strong normalizability $M[x := N']\boldsymbol{L}$ and N' are sn. Hence by IH K is sn.

Case $K = (\lambda x\, M)N\boldsymbol{L}'$ with $L_i \longrightarrow_\beta L_i'$ for i and $L_j = L_j'$ for $j \neq i$. Then we have $M[x := N]\boldsymbol{L} \longrightarrow_\beta M[x := N]\boldsymbol{L}'$. By definition of strong normalizability $M[x := N]\boldsymbol{L}'$ is sn. Hence by IH K is sn. □

Corollary 3.1.6. *If $M[x := N]$ is sc for all sc N, then also $\lambda x\, M$ is sc.*

Proof. Let $M[x := N]$ be sc for all sc N. We have to show that $\lambda x\, M$ is sc. So let N and $\boldsymbol{L}$ be sc such that $(\lambda x\, M)N\boldsymbol{L}$ is of ground type. We must show that $(\lambda x\, M)N\boldsymbol{L}$ is sc. Since $M[x := N]$ is sc by assumption, $M[x := N]\boldsymbol{L}$ is sc too and hence also sn. Since by lemma 3.1.3a N is also sn, by lemma 3.1.5 it follows that $(\lambda x\, M)N\boldsymbol{L}$ is sn and hence also sc. □

Lemma 3.1.7. *Every term M is sc under substitution.*

Proof. By induction on M. *Case* x. The claim follows from lemma 3.1.3b or by assumption.

Case MN. $\boldsymbol{K}$ be sc. We have to show that $M[\boldsymbol{x} := \boldsymbol{K}]N[\boldsymbol{x} := \boldsymbol{K}]$ are sc. This clearly holds, since by IH $M[\boldsymbol{x} := \boldsymbol{K}]$ as well as $N[\boldsymbol{x} := \boldsymbol{K}]$ are sc.

Case $\lambda x\, M$. $\boldsymbol{K}$ be sc. We have to show that $\lambda x\,(M[\boldsymbol{x} := \boldsymbol{K}])$ is sc. We now apply corollary 3.1.6. Let N be sc. By IH for M also $M[x, \boldsymbol{x} := N, \boldsymbol{K}]$ is sc. Hence by corollary 3.1.6 the claim follows. □

From lemma 3.1.7 and lemma 3.1.3 we directly get

Theorem 3.1.8. *$\longrightarrow_\beta$ is terminating, i.e. every term M is sn.* □

Extension of the Termination Proof to Gödel's System T

This proof can easily be extended to terms with pairing $\langle M_0, M_1\rangle$ and projections $\pi_0(M)$, $\pi_1(M)$. We now show that addition of the conversion rules above for Rec does not destroy termination.

We first change the definition of sc for the ground type ι. It is not defined just to be sn (as before), but now is defined inductively, as follows.

Definition 3.1.9. M is sc, if

- all M' such that $M \longrightarrow_\beta M'$ are sc, and in addition
- if $M = \mathrm{S}M_0$, then M_0 is sc.

Theorem 3.1.10. *$\longrightarrow_{\beta\mathsf{Rec}}$ is terminating.*

Proof. We extend the argument above. Clearly it suffices to show that the constants $\mathsf{Rec}_{\iota,\rho}$ and $\mathsf{Rec}_{o,\rho}$ are sc. We restrict ourselves to $\mathsf{Rec}_{\iota,\rho}$; for $\mathsf{Rec}_{o,\rho}$ the argument is similar (and simpler). So let M, N, K be sc. We must show that $\mathsf{Rec}MNK$ is sc.

The proof is by induction on sn for M, N and on sc for K. So let $\boldsymbol{L}$ be sc. We must show that $\mathsf{Rec}MNK\boldsymbol{L}$ is sc. Since this term is not of the form $\mathrm{S}M_0$, it suffices to consider all reducts Q of $\mathsf{Rec}MNK\boldsymbol{L}$ and to show that every such reduct Q is sc.

Case $K = K_0 + 1$, $Q = NK_0(\mathsf{Rec}MNK_0)\boldsymbol{L}$. We first show that $\mathsf{Rec}MNK_0$ is sc. So let $\boldsymbol{L}'$ be sc. Since $K = K_0 + 1$ is sc, by definition also K_0 is. So by induction hypothesis $\mathsf{Rec}MNK_0$ is sc, and we obtain that Q is sc.

If the reduction takes place within a subterm $M, N, K, \boldsymbol{L}$ the claim follows by induction hypothesis (here we need that strong computability is preserved under conversion steps; cf. lemma 3.1.4). Since the case $K = 0$, $Q = M\boldsymbol{L}$ is trivial this proves the claim. □

Now as in section 2.3 we can conclude via Newman's Lemma that the normal form is uniquely determined. For simplicity we identify terms with the same $\beta\mathsf{Rec}$-normal form. Hence every closed term of type ι is identified with a term of the form $S(S(S\ldots(S0)\ldots))$ and every closed term of type o is identified with either true or false. Such terms are denoted by $\underline{n}$ and called *numerals* (even if they are of type o).

3.2 Intuitionistic Arithmetic for Functionals of Finite Type

The system we consider is essentially Heyting's intuitionistic arithmetic in finite types HA^ω as described e.g. in [11]. It is based on Gödel's system T and just adds the corresponding logical and arithmetical apparatus to it. Classical arithmetic is obtained by restriction to formulas without the constructive existential quantifier $\exists^*$ but using the classical definition $\exists = \neg\forall\neg$ instead. Equations are treated on the meta level by identifying terms with the same normal form.

We now write $r, s, t, \ldots$ for (object) terms, and reserve $M, N, K, \ldots$ for derivation terms.

Arithmetic: Predicate symbols and atomic formulas

Every predicate symbol P now has a fixed arity, which is a tupel of types $(\rho_1, \dots, \rho_k)$. So atomic formulas are of the form $P(r_1^{\rho_1}, \dots, r_k^{\rho_k})$. The choice of the predicate symbols depends on the particular problem under consideration. In most cases there will be the predicate symbols $\bot$ (falsity) of arity () and atom of arity (o). The intended interpretation of atom is the set $\{\mathsf{true}\}$. Hence "$\mathsf{atom}(t)$" means "$t = \mathsf{true}$". Every decidable relation (i.e., every relation given by a term t of type o), can then be written in the form $\mathsf{atom}(t)$. If confusion is unlikely, we will write t instead of $\mathsf{atom}(t)$. So formulas are

$$P(r_1^{\rho_1}, \dots, r_k^{\rho_k}) \mid A \wedge B \mid A \to B \mid \forall x^\rho A \mid \exists^* x^\rho A.$$

To accomodate the specific ground types ι and o, we need an induction scheme for both types, and in addition a "truth axiom" .

$$\begin{aligned}
\mathsf{Ind}_{p,A}:&\quad A[p := \mathsf{true}] \to A[p := \mathsf{false}] \to \forall p A\\
\mathsf{Ind}_{n,A}:&\quad A[n := 0] \to (\forall n.A \to A[n := n+1]) \to \forall n A\\
\mathsf{T}:&\quad \mathsf{atom}(\mathsf{true})
\end{aligned}$$

More precisely, in case $\forall p A$ or $\forall n A$ contain free variables (i.e. parameters), these are taken to be universally quantified in front.

Classical logic

Assume that atom is the only predicate symbol, and replace $\bot$ by $\mathsf{atom}(\mathsf{false})$. Then we have

$$\begin{aligned}
\neg A &:= A \to \mathsf{atom}(\mathsf{false})\\
\exists x A &:= \neg \forall x \neg A,
\end{aligned}$$

and stability $\neg\neg A \to A$ is provable for all $\exists^*$-free formulas A, using boolean induction for the case $\mathsf{atom}(t)$, and the stability lemma 1.4.2 thereafter. Hence classical arithmetic (in all finite types) is a subsystem of our present system based on minimal logic.

So for $\exists^*$-free formulas (no restriction from a classical point of view) we have full classical logic.

Note also that for quantifier-free formulas B built from prime formulas of the form $\mathsf{atom}(t)$ we can easily construct a boolean term $t_B : o$ such that

$$\mathsf{atom}(t_B) \leftrightarrow B$$

is derivable. Therefore we can use boolean induction to derive *case distinction* for quantifier-free formulas (cf. (3.15)).

3.3 Program Extraction

We now come back to the full system, and assign to every formula A an object $\tau(A)$ (a type or the symbol $*$). $\tau(A)$ is intended to be the type of the program to be extracted from a proof of A. In case $\tau(A) = *$ proofs of A have no computational content; such formulas A are called *Harrop formulas*.

$$
\begin{aligned}
\tau(P(\boldsymbol{s})) &:= * \\
\tau(\exists^* x^\rho A) &:= \begin{cases} \rho & \text{if } \tau(A) = * \\ \rho \times \tau(A) & \text{otherwise} \end{cases} \\
\tau(\forall x^\rho A) &:= \begin{cases} * & \text{if } \tau(A) = * \\ \rho \to \tau(A) & \text{otherwise} \end{cases} \\
\tau(A_0 \wedge A_1) &:= \begin{cases} \tau(A_i) & \text{if } \tau(A_{1-i}) = * \\ \tau(A_0) \times \tau(A_1) & \text{otherwise} \end{cases} \\
\tau(A \to B) &:= \begin{cases} \tau(B) & \text{if } \tau(A) = * \\ * & \text{if } \tau(B) = * \\ \tau(A) \to \tau(B) & \text{otherwise} \end{cases}
\end{aligned}
$$

We now define, for a given derivation M of a formula A with $\tau(A) \neq *$, its *extracted program* $[\![M]\!]$ of type $\tau(A)$.

$$
\begin{aligned}
[\![u^A]\!] &:= x_u^{\tau(A)} \quad (x_u^{\tau(A)} \text{ uniquely associated with } u^A) \\
[\![\lambda u^A M]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A) = * \\ \lambda x_u^{\tau(A)} [\![M]\!] & \text{otherwise} \end{cases} \\
[\![M^{A \to B} N]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A) = * \\ [\![M]\!][\![N]\!] & \text{otherwise} \end{cases} \\
[\![\langle M_0^{A_0}, M_1^{A_1} \rangle]\!] &:= \begin{cases} [\![M_i]\!] & \text{if } \tau(A_{1-i}) = * \\ \langle [\![M_0]\!], [\![M_1]\!] \rangle & \text{otherwise} \end{cases} \\
[\![\pi_i(M^{A_0 \wedge A_1})]\!] &:= \begin{cases} [\![M]\!] & \text{if } \tau(A_{1-i}) = * \\ \pi_i [\![M]\!] & \text{otherwise} \end{cases} \\
[\![\lambda x^\rho M]\!] &:= \lambda x^\rho [\![M]\!] \\
[\![Mt]\!] &:= [\![M]\!] t
\end{aligned}
$$

We also need extracted programs for induction and $\exists^*$-axioms. For the induction scheme $\mathsf{Ind}_{n,A} \colon A[n := 0] \to (\forall n. A \to A[n := n+1]) \to \forall n A$ let

$$[\![\mathsf{Ind}_{n,A}]\!] := \mathsf{Rec}_{\iota,\rho} \colon \rho \to (\iota \to \rho \to \rho) \to \iota \to \rho,$$

where $\rho = \tau(A)$ (or $\lambda \boldsymbol{x} \mathsf{Rec}_{\iota,\rho}$ if $\boldsymbol{x}$ are the parameters). Similarly boolean induction is realized by $\mathsf{Rec}_{o,\rho}$.

For the $\exists^*$-axioms we set

$$
\begin{aligned}
[\![\exists^{*+}_{x^\rho, A}]\!] &:= \begin{cases} \lambda x^\rho x & \text{if } \tau(A) = * \\ \lambda x^\rho \lambda y^{\tau(A)} \langle x, y \rangle & \text{otherwise} \end{cases} \\
[\![\exists^{*-}_{x^\rho, A(x), B}]\!] &:= \begin{cases} \lambda x^\rho \lambda f^{\rho \to \tau(B)} f x & \text{if } \tau(A) = * \\ \lambda z^{\rho \times \tau(A)} \lambda f^{\rho \to \tau(A) \to \tau(B)} . f \pi_0(z) \pi_1(z) & \text{otherwise} \end{cases}
\end{aligned}
$$

For derivations M^A where $\tau(A) = *$ (i.e. A is a Harrop-formula) we define $[\![M]\!] := \varepsilon$ (ε some new symbol). This applies in particular if A is $\exists^*$-free.

Finally we define the notion of *modified realizability*. More precisely, we define formulas r **mr** A, where A is a formula and r is either a term of type $\tau(A)$ if the latter is a type, or the symbol ε if $\tau(A) = *$.

$$\begin{aligned}
\varepsilon\ \mathbf{mr}\ P(\boldsymbol{s}) &= P(\boldsymbol{s}) \\
r\ \mathbf{mr}\ (\exists^* x A) &= \begin{cases} \varepsilon\ \mathbf{mr}\ A[x := r] & \text{if } \tau(A) = * \\ \pi_1(r)\ \mathbf{mr}\ A[x := \pi_0(r)] & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (\forall x A) &= \begin{cases} \forall x. \varepsilon\ \mathbf{mr}\ A & \text{if } \tau(A) = * \\ \forall x. rx\ \mathbf{mr}\ A & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (A \to B) &= \begin{cases} \varepsilon\ \mathbf{mr}\ A \ \to\ r\ \mathbf{mr}\ B & \text{if } \tau(A) = * \\ \forall x. x\ \mathbf{mr}\ A \ \to\ \varepsilon\ \mathbf{mr}\ B & \text{if } \tau(A) \neq * = \tau(B) \\ \forall x. x\ \mathbf{mr}\ A \ \to\ rx\ \mathbf{mr}\ B & \text{otherwise} \end{cases} \\
r\ \mathbf{mr}\ (A_0 \wedge A_1) &= \begin{cases} \varepsilon\ \mathbf{mr}\ A_0 \ \wedge\ r\ \mathbf{mr}\ A_1 & \text{if } \tau(A_0) = * \\ r\ \mathbf{mr}\ A_0 \ \wedge\ \varepsilon\ \mathbf{mr}\ A_1 & \text{if } \tau(A_1) = * \\ \pi_0(r)\ \mathbf{mr}\ A_0 \ \wedge\ \pi_1(r)\ \mathbf{mr}\ A_1 & \text{otherwise} \end{cases}
\end{aligned}$$

Note that for A $\exists^*$-free we have $\tau(A) = *$ and $\varepsilon\ \mathbf{mr}\ A = A$. For the formulation of the soundness theorem below it will be useful to let $x_u := \varepsilon$ if u^A is an assumption variable with a Harrop-formula A.

Theorem 3.3.1 (Soundness). *If M is a derivation of a formula B, then there is a derivation $\mu(M)$ of $[\![M]\!]\ \mathbf{mr}\ B$ from $\{ x_u\ \mathbf{mr}\ C \mid u^C \in \mathsf{FA}(M) \}$.*

Proof. Induction on M. *Case* $u\colon A$. Then we have $\bar{u}\colon x_u\ \mathbf{mr}\ A$. Let $\mu(u) := \bar{u}$.

Case $\lambda u^A M^B$. We must find a derivation $\mu(\lambda u M)$ of

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B).$$

Subcase $\tau(A) = *$. Then we have $[\![\lambda u M]\!] = [\![M]\!]$, hence

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \varepsilon\ \mathbf{mr}\ A \ \to\ [\![M]\!]\ \mathbf{mr}\ B.$$

By induction hypothesis we can define $\mu(\lambda u M) := \lambda \bar{u} \mu(M)$ with $\bar{u}\colon \varepsilon\ \mathbf{mr}\ A$. *Subcase* $\tau(A) \neq * = \tau(B)$. Then we have $[\![\lambda u M]\!] = \varepsilon$ and

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \forall x. x\ \mathbf{mr}\ A \ \to\ \varepsilon\ \mathbf{mr}\ B,$$

and by induction hypothesis we can define $\mu(\lambda u M) := \lambda x_u \lambda \bar{u} \mu(M)$ with $\bar{u}\colon x_u\ \mathbf{mr}\ A$. *Subcase* $\tau(A) \neq * \neq \tau(B)$. Then we have

$$[\![\lambda u M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \forall x. x\ \mathbf{mr}\ A \ \to\ [\![\lambda u M]\!] x\ \mathbf{mr}\ B$$

Because of $[\![\lambda u M]\!] = \lambda x_u [\![M]\!]$ and since we identify terms with the same β-normal form, again by induction hypothesis we can define $\mu(\lambda u M) := \lambda x_u \lambda \bar{u} \mu(M)$.

Case $M^{A \to B} N^A$. We must find a derivation $\mu(MN)$ of $[\![MN]\!]\ \mathbf{mr}\ B$. *Subcase* $\tau(A) = *$. Then we have $[\![MN]\!] = [\![M]\!]$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \varepsilon\ \mathbf{mr}\ A \ \to\ [\![M]\!]\ \mathbf{mr}\ B$$

and $\mu(N)$ of $\varepsilon\ \mathbf{mr}\ A$; hence we can define $\mu(MN) := \mu(M)\mu(N)$. *Subcase* $\tau(A) \neq * = \tau(B)$. Then we have $[\![MN]\!] = \varepsilon$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \forall x. x\ \mathbf{mr}\ A \ \to\ \varepsilon\ \mathbf{mr}\ B$$

and $\mu(N)$ of $[\![N]\!]\ \mathbf{mr}\ A$; hence we can define $\mu(MN) := \mu(M)[\![N]\!]\mu(N)$. *Subcase* $\tau(A) \neq * \neq \tau(B)$. Then we have $[\![MN]\!] = [\![M]\!][\![N]\!]$. By induction hypothesis we have derivations $\mu(M)$ of

$$[\![M]\!]\ \mathbf{mr}\ (A \to B) \ = \ \forall x. x\ \mathbf{mr}\ A \ \to\ [\![M]\!] x\ \mathbf{mr}\ B$$

and $\mu(N)$ of $[\![N]\!]\ \mathbf{mr}\ A$; hence we can define $\mu(MN) := \mu(M)[\![N]\!]\mu(N)$.

Case $\langle M_0^{A_0}, M_1^{A_1}\rangle$. We must find a derivation $\mu(\langle M_0, M_1\rangle)$ of

$$[\![\langle M_0, M_1\rangle]\!] \;\mathbf{mr}\; (A_0 \wedge A_1).$$

Subcase $\tau(A_0) = * = \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = \varepsilon$, hence

$$[\![\langle M_0, M_1\rangle]\!] \;\mathbf{mr}\; (A_0 \wedge A_1) \quad = \quad \varepsilon \;\mathbf{mr}\; A_0 \;\wedge\; \varepsilon \;\mathbf{mr}\; A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$. *Subcase* $\tau(A_0) = * \neq \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = [\![M_1]\!]$, hence

$$[\![\langle M_0, M_1\rangle]\!] \;\mathbf{mr}\; (A_0 \wedge A_1) \quad = \quad \varepsilon \;\mathbf{mr}\; A_0 \;\wedge\; [\![M_1]\!] \;\mathbf{mr}\; A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$. *Subcase* $\tau(A_0) \neq * = \tau(A_1)$. Similar. *Subcase* $\tau(A_0) \neq * \neq \tau(A_1)$. Then we have $[\![\langle M_0, M_1\rangle]\!] = \langle [\![M_0]\!], [\![M_1]\!]\rangle$, hence

$$[\![\langle M_0, M_1\rangle]\!] \;\mathbf{mr}\; (A_0 \wedge A_1) \quad = \quad [\![M_0]\!] \;\mathbf{mr}\; A_0 \;\wedge\; [\![M_1]\!] \;\mathbf{mr}\; A_1$$

and by induction hypothesis we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_i), \mu(M_{1-i})\rangle$.

Case $\pi_0(M^{A_0 \wedge A_1})$. We must find a derivation $\mu(\pi_0(M))$ of

$$[\![\pi_0(M)]\!] \;\mathbf{mr}\; A_0.$$

Subcase $\tau(A_1) = *$. Then we have $[\![\pi_0(M)]\!] = [\![M]\!]$. by induction hypothesis we have a derivation $\mu(M)$ of

$$[\![M]\!] \;\mathbf{mr}\; (A_0 \wedge A_1) \quad = \quad [\![M]\!] \;\mathbf{mr}\; A_0 \;\wedge\; \varepsilon \;\mathbf{mr}\; A_1.$$

hence we can define $\mu(\pi_0(M)) := \pi_0(\mu(M))$. *Subcase* $\tau(A_0) = * \neq \tau(A_1)$. Then we have $[\![\pi_0(M)]\!] = \varepsilon$. by induction hypothesis we have a derivation $\mu(M)$ of

$$[\![M]\!] \;\mathbf{mr}\; (A_0 \wedge A_1) \quad = \quad \varepsilon \;\mathbf{mr}\; A_0 \;\wedge\; [\![M]\!] \;\mathbf{mr}\; A_1.$$

hence we can define $\mu(\pi_0(M)) := \pi_0(\mu(M))$. *Subcase* $\tau(A_0) \neq * \neq \tau(A_1)$. Similar; we can define $\mu(\langle M_0, M_1\rangle) := \langle \mu(M_0), \mu(M_1)\rangle$.

Case $\pi_1(M^{A_0 \wedge A_1})$. Similar.

Case $\lambda z M^A$. We must find a derivation $\mu(\lambda z M)$ of $[\![\lambda z M]\!] \;\mathbf{mr}\; \forall z A$. By definition $[\![\lambda z M]\!] = \lambda z [\![M]\!]$. *Subcase* $\tau(A) = *$. Then we have

$$\lambda z [\![M]\!] \;\mathbf{mr}\; \forall z A \quad = \quad \forall z.\varepsilon \;\mathbf{mr}\; A$$

and by induction hypothesis we can define $\mu(\lambda z M) := \lambda z \mu(M)$. The variable condition is satisfied, since $\lambda z M^A$ is a derivation term, and hence z does not occur free in any assumption variable $u\colon B$ free in M^A, hence also does not occur free in the free assumption $\bar{u}\colon x_u \;\mathbf{mr}\; B$. *Subcase* $\tau(A) \neq *$. Then we have

$$\lambda z [\![M]\!] \;\mathbf{mr}\; \forall z A \quad = \quad \forall z.(\lambda z [\![M]\!]) z \;\mathbf{mr}\; A.$$

Since we identify terms with the same β-normal form, by induction hypothesis we again can define $\mu(\lambda z M) := \lambda z \mu(M)$. As before one can see that the variable condition is satisfied.

Case $M^{\forall z A} t$. We must find a derivation $\mu(Mt)$ of $[\![Mt]\!] \;\mathbf{mr}\; A[x := t]$. By definition we have $[\![Mt]\!] = [\![M]\!] t$. *Subcase* $\tau(A) = *$. By induction hypothesis we have a derivation of

$$[\![M]\!] \;\mathbf{mr}\; \forall z A \quad = \quad \forall z.\varepsilon \;\mathbf{mr}\; A$$

hence we can define $\mu(Mt) := \mu(M)t$. *Subcase* $\tau(A) \neq *$. By induction hypothesis we have a derivation of

$$[\![M]\!] \;\mathbf{mr}\; \forall z A \quad = \quad \forall z.[\![M]\!] z \;\mathbf{mr}\; A,$$

hence we again can define $\mu(Mt) := \mu(M)t$.

Case $\exists^{*+}_{x,A} : \forall \boldsymbol{x} \forall x. A \to \exists^* x^\rho A$. We must find a derivation $\mu(\exists^{*+}_{x,A})$ of

$$[\![\exists^{*+}_{x,A}]\!] \ \mathbf{mr}\ \forall \boldsymbol{x}\forall x.A \to \exists^* x^\rho A.$$

Subcase $\tau(A) = *$. Then we have $\exists^{*+}_{x,A} = \lambda\boldsymbol{x}\lambda x x$ and we obtain

$$\begin{aligned}(\lambda\boldsymbol{x}\lambda x x)\ \mathbf{mr}\ \forall\boldsymbol{x}\forall x.A \to \exists^* xA &= \forall\boldsymbol{x}\forall x.x\ \mathbf{mr}\ (A \to \exists^* xA)\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to x\ \mathbf{mr}\ \exists^* xA\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ A.\end{aligned}$$

Hence we can define $\mu(\exists^{*+}_{x,A}) = \lambda\boldsymbol{x}\lambda x\lambda u u$. *Subcase* $\tau(A) \neq *$. Then we have $\exists^{*+}_{x,A} = \lambda\boldsymbol{x}\lambda x\lambda y\langle x, y\rangle$ and we obtain

$$\begin{aligned}&\lambda\boldsymbol{x}\lambda x\lambda y\langle x, y\rangle\ \mathbf{mr}\ \forall\boldsymbol{x}\forall x.A \to \exists^* xA\\ &= \forall\boldsymbol{x}\forall x\forall y.y\ \mathbf{mr}\ A \to \langle x, y\rangle\ \mathbf{mr}\ \exists^* xA\\ &= \forall\boldsymbol{x}\forall x\forall y.y\ \mathbf{mr}\ A \to y\ \mathbf{mr}\ A\end{aligned}$$

Hence we can define $\mu(\exists^{*+}_{x,A}) = \lambda\boldsymbol{x}\lambda x\lambda y\lambda u u$.

Case $\exists^{*-}_{x,A,B} : \forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B$ with $x \notin \mathsf{FV}(B)$. We must find a derivation $\mu(\exists^{*-}_{x,A,B})$ of $\forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B$. *Subcase* $\tau(A) = * = \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \varepsilon$ and we obtain

$$\begin{aligned}&\varepsilon\ \mathbf{mr}\ \forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B\\ &= \forall\boldsymbol{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \varepsilon\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ (\forall x.A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall x\,\varepsilon\ \mathbf{mr}\ (A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to (\forall x.\varepsilon\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ B) \to \varepsilon\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\boldsymbol{x}\lambda x\lambda u\lambda v.vxu$. *Subcase* $\tau(A) \neq * = \tau(B)$. Then we again have $[\![\exists^{*-}_{x,A,B}]\!] = \varepsilon$ and we obtain

$$\begin{aligned}&\varepsilon\ \mathbf{mr}\ \forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B\\ &= \forall\boldsymbol{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \varepsilon\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\boldsymbol{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \varepsilon\ \mathbf{mr}\ (\forall x.A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \forall x\,\varepsilon\ \mathbf{mr}\ (A \to B) \to \varepsilon\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to \forall x\forall y(y\ \mathbf{mr}\ A \to \varepsilon\ \mathbf{mr}\ B) \to \varepsilon\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\boldsymbol{x}\lambda x\lambda u\lambda v.v\pi_0(x)\pi_1(x)u$. *Subcase* $\tau(A) = * \neq \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \lambda\boldsymbol{x}\lambda x\lambda z(zx)$ and we obtain

$$\begin{aligned}&\lambda\boldsymbol{x}\lambda x\lambda z(zx)\ \mathbf{mr}\ \forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B\\ &= \forall\boldsymbol{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \lambda z(zx)\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.z\ \mathbf{mr}\ (\forall x.A \to B) \to zx\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.\forall x\ zx\ \mathbf{mr}\ (A \to B) \to zx\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\varepsilon\ \mathbf{mr}\ A \to \forall z.(\forall x.\varepsilon\ \mathbf{mr}\ A \to zx\ \mathbf{mr}\ B) \to zx\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\boldsymbol{x}\lambda x\lambda u\lambda v.vxu$. *Subcase* $\tau(A) \neq * \neq \tau(B)$. Then we have $[\![\exists^{*-}_{x,A,B}]\!] = \lambda\boldsymbol{x}\lambda x\lambda z(z\pi_0(x)\pi_1(x))$ and we obtain

$$\begin{aligned}&\lambda\boldsymbol{x}\lambda x\lambda z(z\pi_0(x)\pi_1(x))\ \mathbf{mr}\ \forall\boldsymbol{x}.\exists^* xA \to (\forall x.A \to B) \to B\\ &= \forall\boldsymbol{x}\forall x.x\ \mathbf{mr}\ \exists^* xA \to \lambda z(z\pi_0(x)\pi_1(x))\ \mathbf{mr}\ ((\forall x.A \to B) \to B)\\ &= \forall\boldsymbol{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to\\ &\qquad \forall z.z\ \mathbf{mr}\ (\forall x.A \to B) \to z\pi_0(x)\pi_1(x)\ \mathbf{mr}\ B\\ &= \forall\boldsymbol{x}\forall x.\pi_1(x)\ \mathbf{mr}\ A[x := \pi_0(x)] \to\\ &\qquad \forall z.\forall x\forall y(y\ \mathbf{mr}\ A \to zxy\ \mathbf{mr}\ B) \to z\pi_0(x)\pi_1(x)\ \mathbf{mr}\ B.\end{aligned}$$

Hence we can define $\mu(\exists^{*-}_{x,A,B}) = \lambda\boldsymbol{x}\lambda x\lambda u\lambda z\lambda v.v\pi_0(x)\pi_1(x)u$. □

If B is $\exists^*$-free, then ε **mr** $B = B$. Hence for $\forall x^\rho \exists^* y^\sigma B$ with $\exists^*$-free B we have $\tau(\forall x \exists y^* B) = \rho \to \sigma$ and

$$t \textbf{ mr } \forall x \exists^* y B = \forall x B[y := tx].$$

Then as a corollary to the soundness theorem we obtain the extraction theorem

Theorem 3.3.2. *From a derivation $M\colon \forall x^\rho \exists^* y^\sigma B$ with B $\exists^*$-free from $\exists^*$-free assumptions Γ one can extract a closed term $[\![M]\!]^{\rho\to\sigma}$ such that the formula $\forall x B[y := [\![M]\!]x]$ is provable from Γ.*

Here Γ should be viewed as lemmata, i.e. true formulas (proved separately, to keep M short). The theorem says that the extracted program is independent of how this shortcut is achieved.

Example: Fibonacci numbers

Let α_n be the n-th Fibonacci number, i.e.

$$\begin{aligned}\alpha_0 &:= 0\\ \alpha_1 &:= 1\\ \alpha_n &:= \alpha_{n-2} + \alpha_{n-1} \quad \text{for } n \geq 2\end{aligned}$$

Here is a naive Scheme-program to compute them

```
(define (f n)
  (if (= n 0)
      0
      (if (= 1 n)
          1
          (+ (f (- n 2)) (f (- n 1))))))
```

Clearly this program is ridiculously inefficient, since it leads to enormous amounts or recomputations. Our goal here is to demonstrate that deriving a program from a constructive proof of the existence of Fibonacci numbers avoids the immediate source of the inefficiency.

We first build a constructive existence proof for the Fibonacci numbers.

```
(add-predicate-constant
  'fib (c-arity 'nat 'nat) "fib")

(aga 'fib0 (pf "fib 0 0"))
(aga 'fib1 (pf "fib 1 1"))
(aga 'fib2 (pf "all n,k,l.fib n k -> fib(n+1)l ->
                                       fib(n+2)(k+l)"))


(set-goal '? (pf "all n ex k fib n k"))
(assume 'n)
(cut (pf "all n ex k,l.fib n k & fib(n+1)l"))

;ok, ?-kernel can be obtained from
;?-kernel-side-premise:
;all n ex k,l.fib n k & fib(n+1)l

;?-kernel-main-premise:
;(all n ex k,l.fib n k & fib(n+1)l) -> ex k fib n k
```

`?-kernel-main-premise` can be proved easily. `?-kernel-side-premise` is proved by induction on n, using `fib0` and `fib1` for the base case and `fib2` for the step case.

The extracted program can now be obtained as follows.

```
(define constr-fibonacci (np pproof))
(define constr-neterm (nt (et constr-fibonacci)))

(lambda (n^1)
  (car ((((nat-rec-at (quote (star nat nat)))
          (cons (num 0) (num 1)))
         (lambda (n^2)
           (lambda (nat*nat^3)
             (cons (cdr nat*nat^3)
                   ((plus-nat (car nat*nat^3))
                    (cdr nat*nat^3)))))) n^1)))
```

A more readable Scheme program is

```
(define (constr-fibo n)
  (car (constr-fibo-aux n)))

(define (constr-fibo-aux n)
  (if (= 0 n)
      (cons 0 1)
      (let ((prev (constr-fibo-aux (- n 1))))
        (cons (cdr prev)
              (+ (car prev) (cdr prev))))))
```

This is a linear algorithm, and values up to `(constr-fibo 5000)` can now be computed. We will see that a classical proof yields a linear algorithm as well, which however uses functions instead of pairs.

Remark 3.3.3. There are other algorithms to compute the Fibonacci numbers α_n, whose relation to some proof is to be explored. In particular, it is not clear whether theses algorithms can be extracted from some reasonable proof. By definition we have

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_{n-2} \\ \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha_{n-1} \\ \alpha_{n-2} + \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha_{n-1} \\ \alpha_n \end{pmatrix},$$

hence with $A = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 1 \end{smallmatrix}\right) \in \mathbb{R}^{2\times 2}$

$$A^n \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_n \\ \alpha_{n+1} \end{pmatrix}.$$

So an easy way to compute the Fibonacci numbers is by computing the powers of A. This can be done in time $O(\log(n))$, since

$$\begin{aligned} A^{2n} &= (A^n)^2 \\ A^{2n+1} &= A^{2n} \cdot A \end{aligned}$$

Moreover, using some elementary linear algebra, one can give an explicit formula for α_n. To this end we first diagonalize the matrix A. The general recipe runs as folows. Form $B := A^t$, hence $B = A$ in our case. The eigenvalues can be computed as the zeros of the characteristic polynomial $p_A = |A - tE| = -t(1-t) - 1 = t^2 - t - 1$. So the eigenvalues are $\lambda_{1,2} = \frac{1\pm\sqrt{5}}{2}$. We now compute eigenvectors for these eigenvalues.

For $\lambda_1 = \frac{1+\sqrt{5}}{2}$. Let $x = \left(\begin{smallmatrix} \xi_1 \\ \xi_2 \end{smallmatrix}\right)$, and solve the linear equation system

$$\begin{pmatrix} -\lambda_1 & 1 \\ 1 & 1-\lambda_1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

So $-\lambda_1\xi_1 + \xi_2$=0. For $\xi_1 = 1$ we obtain $\xi_2 = \lambda_1$, so $x_1 = \left(\begin{smallmatrix} 1 \\ \lambda_1 \end{smallmatrix}\right)$ is an eigenvector. For $\lambda_2 = \frac{1-\sqrt{5}}{2}$ we similarly obtain $x_2 = \left(\begin{smallmatrix} 1 \\ \lambda_2 \end{smallmatrix}\right)$ as an eigenvector.

For the dimensions of the eigenspaces and the multiplicities of the eigenvalues we obtain

$$\begin{aligned} \dim(\mathsf{Eig}(f_B, \lambda_1)) &= 1 = \mu(p_B, \lambda_1) \\ \dim(\mathsf{Eig}(f_B, \lambda_2)) &= 1 = \mu(p_B, \lambda_2), \end{aligned}$$

so the matrix B is diagonalizable, and $\left(\begin{smallmatrix}1\\ \lambda_1\end{smallmatrix}\right), \left(\begin{smallmatrix}1\\ \lambda_2\end{smallmatrix}\right)$ is a basis of $\mathbb{R}^2$ consisting of eigenvectors of f_B. Let

$$T := \begin{pmatrix} 1 & 1 \\ \lambda_1 & \lambda_2 \end{pmatrix}, \qquad S := T^t = \begin{pmatrix} 1 & \lambda_1 \\ 1 & \lambda_2 \end{pmatrix}$$

Then the general theory yields

$$SAS^{-1} = D := \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \qquad S^{-1} = \frac{1}{\lambda_2 - \lambda_1} \begin{pmatrix} \lambda_2 & -\lambda_1 \\ -1 & 2 \end{pmatrix}.$$

We can now give an explicit formula for A^n and hence also for the Fibonacci numbers.

$$\begin{aligned} A^n &= (S^{-1}DS)^n \\ &= \underbrace{(S^{-1}DS) \cdot \ldots \cdot (S^{-1}DS)}_{n \text{ times}} \\ &= S^{-1}D^nS \\ &= S^{-1} \begin{pmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{pmatrix} S \\ &= S^{-1} \begin{pmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{pmatrix} \begin{pmatrix} 1 & \lambda_1 \\ 1 & \lambda_2 \end{pmatrix} \\ &= S^{-1} \begin{pmatrix} \lambda_1^n & \lambda_1^{n+1} \\ \lambda_2^n & \lambda_2^{n+1} \end{pmatrix} \\ &= \frac{1}{\lambda_2 - \lambda_1} \begin{pmatrix} \lambda_2 & -\lambda_1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} \lambda_1^n & \lambda_1^{n+1} \\ \lambda_2^n & \lambda_2^{n+1} \end{pmatrix} \\ &= -\frac{1}{\sqrt{5}} \begin{pmatrix} \lambda_1^n\lambda_2 - \lambda_2^n\lambda_1 & \lambda_2\lambda_1^{n+1} - \lambda_1\lambda_2^{n+1} \\ -\lambda_1^n + \lambda_2^n & -\lambda_1^{n+1} + \lambda_2^{n+1} \end{pmatrix} \\ &= -\frac{1}{\sqrt{5}} \begin{pmatrix} -\lambda_1^{n+1} + \lambda_2^{n+1} & -\lambda_1^n + \lambda_2^n \\ -\lambda_1^n + \lambda_2^n & -\lambda_1^{n+1} + \lambda_2^{n+1} \end{pmatrix} && \text{since } \lambda_1\lambda_2 = -1 \\ &= \frac{1}{\sqrt{5}} \begin{pmatrix} \beta_{n-1} & \beta_n \\ \beta_n & \beta_{n+1} \end{pmatrix} && \text{with } \beta_n := \lambda_1^n - \lambda_2^n. \end{aligned}$$

Hence by the above

$$\begin{pmatrix} \alpha_n \\ \alpha_{n+1} \end{pmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} \beta_{n-1} & \beta_n \\ \beta_n & \beta_{n+1} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

In particular we then have for α_n

$$\alpha_n = \frac{1}{\sqrt{5}}\beta_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right].$$

Other examples for program extraction from constructive proofs are abundant in the liteature. Major case studies done in our group include the development of the Warshall algorithm in [6], and of the Dijkstra algorithm in [3].

3.4 *A*-translation: Computational Content of Classical Proofs

For simplicity we only treat formulas in the $\to\forall$ fragment; this is not an essential restriction, since conjunctions can be easily removed (cf. lemma 1.7.3).

A formula is *relevant* if it "ends" with $\bot$. More precisely, relevant formulas are defined inductively by the clauses

– $\bot$ is relevant,
– if C is relevant and B is an arbitrary formula, then $B \to C$ is relevant, and
– if C is relevant, then $\forall x C$ is relevant.

A formula which is not relevant is called *irrelevant*.

We define *goal formulas* G and *definite formulas* D inductively. These notions are related to similar ones common under the same name in the context of extensions of logic programming. Let P range over atoms (including $\bot$).

$$\begin{array}{rcll} G & := & P \mid D \to G & \text{provided } D \text{ irrelevant} \Rightarrow D \text{ quantifier-free} \\ & & \mid \forall x G & \text{provided } G \text{ irrelevant} \\[1ex] D & := & P \mid G \to D & \text{provided } D \text{ irrelevant} \Rightarrow G \text{ irrelevant} \\ & & \mid \forall x D & \end{array}$$

Let $\mathcal{A}$ be a predicate symbol and write $C^{\mathcal{A}}$ for $C[\bot := \mathcal{A}]$.

Lemma 3.4.1. *For definite formulas D and goal formulas G we have derivations*

$$(\neg D \to \mathcal{A}) \to D^{\mathcal{A}} \qquad \textit{for } D \textit{ relevant} \tag{3.4}$$

$$D \to D^{\mathcal{A}} \tag{3.5}$$

$$G^{\mathcal{A}} \to G \qquad \textit{for } G \textit{ irrelevant} \tag{3.6}$$

$$G^{\mathcal{A}} \to (G \to \mathcal{A}) \to \mathcal{A} \qquad \textit{for } G \textit{ relevant} \tag{3.7}$$

from

$$\begin{array}{ll} \bot \to \mathcal{A} & \textit{ex-falso for } \mathcal{A} \\ (D \to \mathcal{A}) \to (\neg D \to \mathcal{A}) \to \mathcal{A} & \textit{cases for } \mathcal{A}, \textit{ with quantifier-free } D. \end{array}$$

Note that by strengthening the hypothesis $G \to \mathcal{A}$ of (3.7) to $\neg G$ we obtain

$$G^{\mathcal{A}} \to \neg G \to \mathcal{A} \qquad \textit{for } G \textit{ relevant,} \tag{3.8}$$

again using $\bot \to \mathcal{A}$.

Proof. Simultaneous induction on formulas.

(3.4). Let D be relevant. *Case* $\bot$. Clearly $((\bot \to \bot) \to \mathcal{A}) \to \mathcal{A}$ is derivable. *Case* $G \to D$. *Subcase* G relevant.

$$\dfrac{\dfrac{\begin{matrix} | \\ (\neg D \to \mathcal{A}) \to D^{\mathcal{A}} \end{matrix} \quad \dfrac{\dfrac{\dfrac{\begin{matrix} | \\ G^{\mathcal{A}} \to (G \to \mathcal{A}) \to \mathcal{A} \end{matrix} \quad G^{\mathcal{A}}}{(G \to \mathcal{A}) \to \mathcal{A}} \quad \dfrac{\dfrac{\neg(G \to D) \to \mathcal{A} \quad \dfrac{\dfrac{\neg D \quad \dfrac{G \to D \quad G}{D}}{\bot}}{\neg(G \to D)}}{\mathcal{A}}}{G \to \mathcal{A}}}{\mathcal{A}}}{\neg D \to \mathcal{A}}}{D^{\mathcal{A}}}}{(\neg(G \to D) \to \mathcal{A}) \to G^{\mathcal{A}} \to D^{\mathcal{A}}}$$

Here we have used the induction hypotheses (3.4) for D and (3.7) for G. *Subcase* G irrelevant.

$$\dfrac{\dfrac{\overset{|}{(\neg D\to\mathcal{A})\to D^{\mathcal{A}}} \qquad \dfrac{\dfrac{\neg(G\to D)\to\mathcal{A} \qquad \dfrac{\dfrac{\neg D \qquad \dfrac{G\to D \qquad \dfrac{\overset{|}{G^{\mathcal{A}}\to G} \qquad G^{\mathcal{A}}}{G}}{D}}{\bot}}{\neg(G\to D)}}{\mathcal{A}}}{\neg D\to\mathcal{A}}}{D^{\mathcal{A}}}}{(\neg(G\to D)\to\mathcal{A})\to G^{\mathcal{A}}\to D^{\mathcal{A}}}$$

Here we have used the induction hypotheses (3.4) for D and (3.6) for G. *Case* $\forall x D$.

$$\dfrac{\dfrac{\dfrac{\overset{|}{(\neg D\to\mathcal{A})\to D^{\mathcal{A}}} \qquad \dfrac{\dfrac{\neg\forall x D\to\mathcal{A} \qquad \dfrac{\dfrac{\neg D \qquad \dfrac{\forall x D}{D}}{\bot}}{\neg\forall x D}}{\mathcal{A}}}{\neg D\to\mathcal{A}}}{D^{\mathcal{A}}}}{\forall x D^{\mathcal{A}}}}{(\neg\forall x D\to\mathcal{A})\to\forall x D^{\mathcal{A}}}$$

Here we have used the induction hypothesis (3.4) for D.

(3.5). *Case* D relevant.

$$\dfrac{\dfrac{\overset{|}{(\neg D\to\mathcal{A})\to D^{\mathcal{A}}} \qquad \dfrac{\dfrac{\bot\to\mathcal{A} \qquad \dfrac{\neg D \qquad D}{\bot}}{\mathcal{A}}}{\neg D\to\mathcal{A}}}{D^{\mathcal{A}}}}{D\to D^{\mathcal{A}}}$$

Here we have used (3.4) and $\bot\to\mathcal{A}$.

Case D irrelevant. *Subcase* P. Then $P^{\mathcal{A}} = P$ and the claim is obvious. *Subcase* $G\to D$. Then D is irrelevant, hence also G is irrelevant.

$$\dfrac{\dfrac{\overset{|}{D\to D^{\mathcal{A}}} \qquad \dfrac{G\to D \qquad \dfrac{\overset{|}{G^{\mathcal{A}}\to G} \qquad G^{\mathcal{A}}}{G}}{D}}{D^{\mathcal{A}}}}{(G\to D)\to G^{\mathcal{A}}\to D^{\mathcal{A}}}$$

Here we have used the induction hypotheses (3.6) for G and (3.5) for D. *Subcase* $\forall x D$

$$\dfrac{\dfrac{\dfrac{\overset{|}{D\to D^{\mathcal{A}}} \qquad \dfrac{\forall x D}{D}}{D^{\mathcal{A}}}}{\forall x D^{\mathcal{A}}}}{\forall x D\to\forall x D^{\mathcal{A}}}$$

Here we have used the induction hypothesis (3.5) for D.

(3.6). Let G be irrelevant. *Subcase* P. Then $P^{\mathcal{A}} = P$ and the claim is obvious. *Subcase* $D\to G$.

$$\dfrac{\dfrac{\begin{matrix}|\\ G^{\mathcal{A}}\to G\end{matrix} \qquad \dfrac{D^{\mathcal{A}}\to G^{\mathcal{A}} \qquad \dfrac{\begin{matrix}|\\ D\to D^{\mathcal{A}}\end{matrix}\qquad D}{D^{\mathcal{A}}}}{G^{\mathcal{A}}}}{G}}{(D^{\mathcal{A}}\to G^{\mathcal{A}})\to D\to G}$$

Here we have used the induction hypotheses (3.6) for G and (3.5) for D. *Case* $\forall xG$.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}|\\ G^{\mathcal{A}}\to G\end{matrix}\qquad \dfrac{\forall xG^{\mathcal{A}}}{G^{\mathcal{A}}}}{G}}{\forall xG}}{\forall xG^{\mathcal{A}}\to\forall xG}$$

Here we have used the induction hypothesis (3.6) for G.

(3.7). Let G be relevant. *Case* $\bot$. Obvious, since $\bot^{\mathcal{A}}=\mathcal{A}$. *Case* $D\to G$. *Subcase* D relevant.

$$\dfrac{\dfrac{\dfrac{\begin{matrix}|\\ G^{\mathcal{A}}\to(G\to\mathcal{A})\to\mathcal{A}\end{matrix}\qquad \dfrac{D^{\mathcal{A}}\to G^{\mathcal{A}}\qquad \dfrac{\begin{matrix}|\\ (\neg D\to\mathcal{A})\to D^{\mathcal{A}}\end{matrix}\qquad \dfrac{\dfrac{(D\to G)\to\mathcal{A}\qquad \dfrac{\dfrac{\dfrac{\neg D\quad D}{\bot}}{G}}{D\to G}}{\mathcal{A}}}{\neg D\to\mathcal{A}}}{D^{\mathcal{A}}}}{G^{\mathcal{A}}}}{(G\to\mathcal{A})\to\mathcal{A}}\qquad \dfrac{\dfrac{(D\to G)\to\mathcal{A}\qquad \dfrac{G}{D\to G}}{\mathcal{A}}}{G\to\mathcal{A}}}{\mathcal{A}}}{(D^{\mathcal{A}}\to G^{\mathcal{A}})\to((D\to G)\to\mathcal{A})\to\mathcal{A}}$$

Here we have used the induction hypotheses (3.7) for G and (3.4) for D. Note that the passage from $\bot$ to G can be done by means of introduction rules, since G is relevant.

Subcase D irrelevant. Then D is quantifier-free. We use case distinction on D. In the positive case we have

$$\dfrac{\dfrac{\dfrac{\begin{matrix}|\\ G^{\mathcal{A}}\to(G\to\mathcal{A})\to\mathcal{A}\end{matrix}\qquad \dfrac{D^{\mathcal{A}}\to G^{\mathcal{A}}\qquad \dfrac{\begin{matrix}|\\ D\to D^{\mathcal{A}}\end{matrix}\quad D}{D^{\mathcal{A}}}}{G^{\mathcal{A}}}}{(G\to\mathcal{A})\to\mathcal{A}}\qquad \dfrac{\dfrac{(D\to G)\to\mathcal{A}\qquad \dfrac{G}{D\to G}}{\mathcal{A}}}{G\to\mathcal{A}}}{\mathcal{A}}}{D\to\mathcal{A}}$$

Here we have used the induction hypotheses (3.7) for G and (3.5) for D. In the negative case we obtain

$$\dfrac{\dfrac{(D\to G)\to\mathcal{A}\qquad \dfrac{\dfrac{\dfrac{\neg D\qquad D}{\bot}}{G}}{D\to G}}{\mathcal{A}}}{\neg D\to\mathcal{A}}$$

Again the passage from $\bot$ to G can be done by means of introduction rules, since G is relevant. Now the cases scheme $(D\to\mathcal{A})\to(\neg D\to\mathcal{A})\to\mathcal{A}$ yields $\mathcal{A}$, hence we have derived

$$(D^{\mathcal{A}}\to G^{\mathcal{A}})\to((D\to G)\to\mathcal{A})\to\mathcal{A}. \qquad \square$$

Lemma 3.4.2. *For goal formulas* $\boldsymbol{G}$ *we have a derivation*

$$(\boldsymbol{G} \to \mathcal{A}) \to \boldsymbol{G}^{\mathcal{A}} \to \mathcal{A}$$

from the assumptions in lemma 3.4.1, and additionally

$$(G_j \to \mathcal{A}) \to (\neg G_j \to \mathcal{A}) \to \mathcal{A} \quad \textit{for the relevant } G_j.$$

Proof. For ease in notation let $\boldsymbol{G} = G_1, G_2$ with G_1 irrelevant and G_2 relevant.

$$\dfrac{\dfrac{G_1 \to G_2 \to \mathcal{A} \qquad \dfrac{\dfrac{\mid}{G_1^{\mathcal{A}} \to G_1} \qquad G_1^{\mathcal{A}}}{G_1}}{G_2 \to \mathcal{A}} \qquad \dfrac{\dfrac{\mid}{G_2^{\mathcal{A}} \to \neg G_2 \to \mathcal{A}} \qquad G_2^{\mathcal{A}}}{\neg G_2 \to \mathcal{A}}}{\mathcal{A}}$$

Here we have used (3.6) and (3.8), and cases for G_2. □

Theorem 3.4.3. *Assume that for definite formulas* $\boldsymbol{D}$ *and goal formulas* $\boldsymbol{G}$ *we have a derivation of*

$$\boldsymbol{D} \to (\forall y.\boldsymbol{G} \to \bot) \to \bot.$$

Then we can construct a derivation of

$$\boldsymbol{D} \to (\forall y.\boldsymbol{G} \to \mathcal{A}) \to \mathcal{A}$$

from

$$\begin{array}{ll} \bot \to \mathcal{A} & \\ (D \to \mathcal{A}) \to (\neg D \to \mathcal{A}) \to \mathcal{A} & \textit{for some quantifier-free } D \\ (G_j \to \mathcal{A}) \to (\neg G_j \to \mathcal{A}) \to \mathcal{A} & \textit{for the relevant } G_j. \end{array}$$

Proof. Substitution of $\mathcal{A}$ for $\bot$ in the given derivation yields a derivation of

$$\boldsymbol{D}^{\mathcal{A}} \to (\forall y.\boldsymbol{G}^{\mathcal{A}} \to \mathcal{A}) \to \mathcal{A}.$$

Now lemma 3.4.1 allows to drop $\mathcal{A}$ in $D^{\mathcal{A}}$ and lemma 3.4.2 allows to drop $\mathcal{A}$ in $\boldsymbol{G}^{\mathcal{A}}$. □

Recall that $\mathcal{A}$ is an arbitrary predicate symbol, and that up to this point we have not used the strong existential quantifier $\exists^*$. Now substituting $\exists^* y\boldsymbol{G}$ for $\mathcal{A}$ makes the premise $\forall y.\boldsymbol{G} \to \exists^* y\boldsymbol{G}$ obviously provable and we obtain

Corollary 3.4.4. *Assume that for definite formulas* $\boldsymbol{D}$ *and goal formulas* $\boldsymbol{G}$ *we have a derivation of*

$$\boldsymbol{D} \to (\forall y.\boldsymbol{G} \to \bot) \to \bot.$$

Then we can construct a derivation of

$$\boldsymbol{D} \to \exists^* y\boldsymbol{G}$$

from

$$\begin{array}{ll} \bot \to \exists^* y\boldsymbol{G} & \\ (D \to \exists^* y\boldsymbol{G}) \to (\neg D \to \exists^* y\boldsymbol{G}) \to \exists^* y\boldsymbol{G} & \textit{for some quantifier-free } D \\ (G_j \to \exists^* y\boldsymbol{G}) \to (\neg G_j \to \exists^* y\boldsymbol{G}) \to \exists^* y\boldsymbol{G} & \textit{for the relevant } G_j. \quad \square \end{array}$$

Computational Content

Since we have been rather explicit in our constructions of derivations, it is possible to spell out their computational content (i.e. their extracted programs) also rather explicitly. We will demonstrate this in some examples below, but first let us give the general statement. To achieve this, we go through the lemmata again, this time taking computational content into account.

Lemma 3.4.5. *For definite formulas D, goal formulas G and an arbitrary formula A we have derivations*

$$(\neg D \to x \mathbf{\ mr\ } A) \to [\![M_D^A]\!](x) \mathbf{\ mr\ } D^A \qquad \text{for } D \text{ relevant} \tag{3.9}$$
$$D \to [\![N_D^A]\!] \mathbf{\ mr\ } D^A \qquad \text{for } D \text{ relevant} \tag{3.10}$$
$$D \to \varepsilon \mathbf{\ mr\ } D^A \qquad \text{for } D \text{ irrelevant} \tag{3.11}$$
$$\varepsilon \mathbf{\ mr\ } G^A \to G \qquad \text{for } G \text{ irrelevant} \tag{3.12}$$
$$x \mathbf{\ mr\ } G^A \to (G \to y \mathbf{\ mr\ } A) \to [\![L_G^A]\!](x,y) \mathbf{\ mr\ } A \qquad \text{for } G \text{ relevant} \tag{3.13}$$
$$x \mathbf{\ mr\ } G^A \to \neg G \to [\![H_G^A]\!](x) \mathbf{\ mr\ } A \qquad \text{for } G \text{ relevant.} \tag{3.14}$$

Proof. Take the derivations from lemma 3.4.1 and substitute the predicate symbol $\mathcal{A}$ by the formula A. Then the assumptions

$$\begin{array}{ll} \bot \to \mathcal{A} & \text{ex-falso for } \mathcal{A} \\ (D \to \mathcal{A}) \to (\neg D \to \mathcal{A}) \to \mathcal{A} & \text{cases for } \mathcal{A}\text{, with quantifier-free } D \end{array}$$

used in these derivations are transformed into derivable formulas (to see this, recall that our only atoms are now of the form $\mathsf{atom}(t)$ with a boolean term t, and we have boolean induction as well as the truth axiom $\mathsf{atom}(\mathsf{true})$ available). Plugging these derivations into the ones we have yields

$$\begin{array}{rll} M_D^A\colon & (\neg D \to A) \to D^A & \text{for } D \text{ relevant} \\ N_D^A\colon & D \to D^A & \\ K_G^A\colon & G^A \to G & \text{for } G \text{ irrelevant} \\ L_G^A\colon & G^A \to (G \to A) \to A & \text{for } G \text{ relevant} \\ H_G^A\colon & G^A \to \neg G \to A & \text{for } G \text{ relevant.} \end{array}$$

Now the soundness theorem 3.3.1 gives the claim. For example, in case M_D^A we argue as follows.

$$\begin{array}{l} M_D^A\colon (\neg D \to A) \to D^A \\ [\![M_D^A]\!] \mathbf{\ mr\ } \big((\neg D \to A) \to D^A\big) \\ \forall x. x \mathbf{\ mr\ } (\neg D \to A) \to [\![M_D^A]\!](x) \mathbf{\ mr\ } D^A \\ \forall x. (\neg D \to x \mathbf{\ mr\ } A) \to [\![M_D^A]\!](x) \mathbf{\ mr\ } D^A. \end{array}$$

In the other cases the argument is similar. □

Lemma 3.4.6. *Let $G_1, \dots, G_k$ be relevant quantifier-free and $G_{k+1}, \dots, G_m$ irrelevant goal formulas, $A := \exists^* y \boldsymbol{G}$ and let $\boldsymbol{x} \mathbf{\ mr\ } \boldsymbol{G}^A \to B$ abbreviate*

$$x_1 \mathbf{\ mr\ } G_1^A \to \dots \to x_k \mathbf{\ mr\ } G_k^A \to \varepsilon \mathbf{\ mr\ } G_{k+1}^A \to \dots \to \varepsilon \mathbf{\ mr\ } G_m^A \to B.$$

Let $\rho := \tau(A)$ (i.e. the type of y) and

$$\begin{aligned} t :&= \lambda y, \boldsymbol{x}. \Phi_\rho(\neg G_1, [\![H_{G_1}^A]\!](x_1), \dots \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y) \dots) \\ &= \lambda y, \boldsymbol{x}. \textbf{\textit{if}}\ \neg G_1\ \textbf{\textit{then}}\ [\![H_{G_1}^A]\!](x_1)\ \textbf{\textit{else}} \dots\ \textbf{\textit{if}}\ \neg G_k\ \textbf{\textit{then}}\ [\![H_{G_k}^A]\!](x_k)\ \textbf{\textit{else}}\ y \end{aligned}$$

Then

$$\vdash \forall y, \boldsymbol{x}. \boldsymbol{x} \mathbf{\ mr\ } \boldsymbol{G}^A \to t y \boldsymbol{x} \mathbf{\ mr\ } A.$$

Proof. Recall that for quantifier-free formulas B built from prime formulas of the form $\mathsf{atom}(t)$ we can construct a boolean term $t_B\colon o$ such that $\mathsf{atom}(t_B) \leftrightarrow B$ is derivable. Writing $\Phi_\sigma(B, x^\sigma, y^\sigma)$ for **if** t_B **then** x **else** y we can therefore derive from boolean induction

$$(B \to x \text{ mr } C) \to (\neg B \to y \text{ mr } C) \to \Phi_\sigma(B, x, y) \text{ mr } C \tag{3.15}$$

with $\sigma = \tau(C)$.

Using the fact that $\boldsymbol{G} \to y \text{ mr } A$ is obviously derivable given our choice of $A := \exists^* y\boldsymbol{G}$, and that for all $i = k+1, \ldots, m$ (i.e. for G_i irrelevant) we have the derivations

$$\dfrac{\begin{array}{c}| \\ \varepsilon \text{ mr } G_i^A \to G_i\end{array} \qquad \varepsilon \text{ mr } G_i^A}{G_i}$$

we obtain a derivation of

$$G_k \to G_{k-1} \to \cdots \to G_1 \to y \text{ mr } A$$

from the assumption $\varepsilon \text{ mr } G_{k+1}^A, \ldots, \varepsilon \text{ mr } G_m^A$.

For $i = 1, \ldots, k$ (i.e. for G_i relevant) we have

$$\dfrac{\begin{array}{c}| \\ x_i \text{ mr } G_i^A \to \neg G_i \to [\![H_{G_i}^A]\!](x_i) \text{ mr } A\end{array} \qquad x_i \text{ mr } G_i^A}{\neg G_i \to [\![H_{G_i}^A]\!](x_i) \text{ mr } A}$$

Using (3.15) we obtain

$$\begin{aligned}
&(\neg G_k \to [\![H_{G_k}^A]\!](x_k) \text{ mr } A) \to \\
&(G_k \to G_{k-1} \to \cdots \to G_1 \to y \text{ mr } A) \to \\
&G_{k-1} \to \cdots \to G_1 \to \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y) \text{ mr } A,
\end{aligned}$$

similarly

$$\begin{aligned}
&(\neg G_{k-1} \to [\![H_{G_{k-1}}^A]\!](x_{k-1}) \text{ mr } A) \to \\
&(G_{k-1} \to \cdots \to G_1 \to \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y) \text{ mr } A) \to \\
&G_{k-2} \to \cdots \to G_1 \to \Phi_\rho(\neg G_{k-1}, [\![H_{G_{k-1}}^A]\!](x_{k-1}), \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y)) \text{ mr } A
\end{aligned}$$

and finally

$$\begin{aligned}
&(\neg G_1 \to [\![H_{G_1}^A]\!](x_1) \text{ mr } A) \to \\
&(G_1 \to \Phi_\rho(\neg G_2, [\![H_{G_2}^A]\!](x_2), \ldots \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y) \ldots) \text{ mr } A) \to \\
&\Phi_\rho(\neg G_1, [\![H_{G_1}^A]\!](x_1), \Phi_\rho(\neg G_2, [\![H_{G_2}^A]\!](x_2), \ldots \Phi_\rho(\neg G_k, [\![H_{G_k}^A]\!](x_k), y) \ldots)) \text{ mr } A
\end{aligned}$$

Plugging all these derivations together we obtain

$$\boldsymbol{x} \text{ mr } \boldsymbol{G}^A \to ty\boldsymbol{x} \text{ mr } A,$$

which implies the claim. □

Theorem 3.4.7. *Let $D_1, \ldots, D_l$ be relevant, $D_{l+1}, \ldots, D_n$ irrelevant definite formulas, and $G_1, \ldots, G_k$ be relevant quantifier-free and $G_{k+1}, \ldots, G_m$ irrelevant goal formulas. Let $A := \exists^* y\boldsymbol{G}$ and*

$$t := \lambda y, \boldsymbol{x}.\textit{\textbf{if}}\ \neg G_1\ \textit{\textbf{then}}\ [\![H_{G_1}^A]\!](x_1)\ \textit{\textbf{else}}\ \ldots\ \textit{\textbf{if}}\ \neg G_k\ \textit{\textbf{then}}\ [\![H_{G_k}^A]\!](x_k)\ \textit{\textbf{else}}\ y.$$

Assume that we have a derivation

$$M\colon \boldsymbol{D} \to (\forall y.\boldsymbol{G} \to \bot) \to \bot.$$

Then we can derive

$$\boldsymbol{D} \to [\![M^A]\!][\![N_{D_1}^A]\!] \ldots [\![N_{D_l}^A]\!]t \text{ mr } \exists^* y\boldsymbol{G}.$$

Proof. Since the given derivation takes place in minimal logic we can substitute $\bot$ by A everywhere and obtain

$$M^A : \boldsymbol{D}^A \to (\forall y. \boldsymbol{G}^A \to A) \to A.$$

The soundness theorem 3.3.1 yields

$$\begin{aligned}&[\![M^A]\!] \;\mathbf{mr}\; (\boldsymbol{D}^A \to (\forall y.\boldsymbol{G}^A \to A) \to A)\\ &\forall \boldsymbol{u}, v. \boldsymbol{u} \;\mathbf{mr}\; \boldsymbol{D}^A \to (\forall y, \boldsymbol{x}. \boldsymbol{x} \;\mathbf{mr}\; \boldsymbol{G}^A \to vy\boldsymbol{x} \;\mathbf{mr}\; A) \to [\![M^A]\!](\boldsymbol{u}, v) \;\mathbf{mr}\; A,\end{aligned}$$

where $\boldsymbol{u} \;\mathbf{mr}\; \boldsymbol{D}^A \to B$ abbreviates

$$u_1 \;\mathbf{mr}\; D_1^A \to \cdots \to u_l \;\mathbf{mr}\; D_l^A \to \varepsilon \;\mathbf{mr}\; D_{l+1} \to \cdots \to \varepsilon \;\mathbf{mr}\; D_n \to B.$$

Now substitute $[\![N_{D_1}^A]\!], \dots, [\![N_{D_l}^A]\!]$ for $u_1, \dots, u_l$ and t for v. Then the premises

$$[\![N_{D_1}^A]\!] \;\mathbf{mr}\; D_1^A, \dots, [\![N_{D_l}^A]\!] \;\mathbf{mr}\; D_l^A, \varepsilon \;\mathbf{mr}\; D_{l+1}, \dots, \varepsilon \;\mathbf{mr}\; D_n$$

are derivable from $\boldsymbol{D}$ by (3.10) and (3.11), and the premise $\boldsymbol{x} \;\mathbf{mr}\; \boldsymbol{G}^A \to ty\boldsymbol{x} \;\mathbf{mr}\; A$ is derivable by lemma 3.4.6. Therefore we obtain

$$\boldsymbol{D} \to [\![M^A]\!][\![N_{D_1}^A]\!] \dots [\![N_{D_n}^A]\!] t \;\mathbf{mr}\; A$$

and hence the claim by definition of **mr** . □

How to obtain definite and goal formulas

To apply these results we have to know that our assumptions are definite formulas and our goal is given by goal formulas. Clearly this can always be achieved by inserting double negations in front of every atom (cf. the definitions of definite and goal formulas). This corresponds to the original (unrefined) so-called A-translation of Friedman [14]. However, in order to obtain reasonable programs which do not unneccessarily use higher types we want to insert double negations only at as few places as possible.

We describe a general way to obtain definite and goal formulas, following [5]. It consists in singling out some predicate symbols as being "critical", and then double negating only the atoms formed with critical predicate symbols; call these *critical* atoms.

Assume we have a proof of

$$\forall \boldsymbol{x}_1 C_1 \to \cdots \to \forall \boldsymbol{x}_n C_n \to (\forall \boldsymbol{y}. \boldsymbol{B} \to \bot) \to \bot$$

with $\boldsymbol{C}, \boldsymbol{B}$ quantifier-free. Let

$$L := \{ C_1, \dots, C_n, \boldsymbol{B} \to \bot \}$$

The set of L-*critical* predicate symbols is defined to be the smallest set satisfying

(i) $\bot$ is critical.
(ii) If $(\boldsymbol{C}_1 \to P_1(\boldsymbol{s}_1)) \to \cdots \to (\boldsymbol{C}_m \to P_m(\boldsymbol{s}_m)) \to R(\boldsymbol{t})$ is a positive subformula of L, and if some P_i is L-critical, then R is L-critical.

Now if we double negate every L-critical atom we clearly obtain definite assumptions $\boldsymbol{C}'$ and goal formulas $\boldsymbol{B}'$.

However, we may be able to obtain definite and goal formulas with still fewer double negations: it may not be necessary to double negate *every* critical atom.

3.5 Examples

Fibonacci Numbers Again

We come back to our initial example of a proof of the existence of Fibonacci numbers, but this time based on a classical proof. It will turn out that we obtain a much better program, where we pass λ-expressions rather than pairs. We want to derive

$$(\forall k.F(n,k) \to \bot) \to \bot$$

from assumptions expressing that F is the graph of the Fibonacci function, i.e.

$$F(0,0),$$
$$F(1,1),$$
$$\forall n,k,\ell.F(n,k) \to F(n+1,\ell) \to F(n+2,k+\ell).$$

Clearly the assumption formulas are definite and $F(n,k)$ is a goal formula. So theorem 3.4.7 can be applied without inserting double negations.

To construct a derivation, assume

$$\begin{aligned} v_0 &: F(0,0), \\ v_1 &: F(1,1), \\ v_2 &: \forall n,k,l.F(n,k) \to F(n+1,\ell) \to F(n+2,k+\ell) \\ u &: \forall k.F(n,k) \to \bot. \end{aligned}$$

Our goal is $\bot$. To this end we first prove a strengthened claim in order to get the induction through:

$$\forall n B \quad \text{with } B := (\forall k,\ell.F(n,k) \to F(n+1,\ell) \to \bot) \to \bot.$$

This is proved by induction on n. The base case follows from v_0 and v_1. In the step case we can assume that we have k,ℓ satisfying $F(n,k)$ and $F(n+1,\ell)$. We need k',ℓ' such that $F(n+1,k') \to F(n+2,\ell')$. Using v_2 simply take $k' := \ell$ and $\ell' := k+\ell$. – To obtain our goal $\bot$ from $\forall nB$, it clearly suffices to prove its premise $\forall k,\ell.F(n,k) \to F(n+1,\ell) \to \bot$. So let k,ℓ be given and assume $u_1 : F(n,k)$ and $u_2 : F(n+1,\ell)$. Then u applied to k and u_1 gives our goal $\bot$.

The derivation term is

$$\begin{aligned} M =&\lambda v_0^{F(0,0)} \lambda v_1^{F(1,1)} \lambda v_2^{\forall n,k,\ell.F(n,k)\to F(n+1,\ell)\to F(n+2,k+\ell)} \lambda u^{\forall k.F(n,k)\to\bot} \\ &\mathsf{Ind}_{n,B} M_{\text{base}} M_{\text{step}} n (\lambda k \lambda \ell \lambda u_1^{F(n,k)} \lambda u_2^{F(n+1,\ell)}.uku_1) \end{aligned}$$

where

$$\begin{aligned} M_{\text{base}} =&\lambda w_0^{\forall k,\ell.F(0,k)\to F(1,\ell)\to\bot}.w_0 01 v_0 v_1 \\ M_{\text{step}} =&\lambda n \lambda w^B \lambda w_1^{\forall k,\ell.F(n+1,k)\to F(n+2,\ell)\to\bot}. \\ &w(\lambda k \lambda \ell \lambda u_3^{F(n,k)} \lambda u_4^{F(n+1,\ell)}.w_1 \ell (k+\ell) u_4 (v_2 k \ell u_3 u_4)) \end{aligned}$$

Now let $A := \exists^* k F(n,k)$, and M^A be obtained from M by replacing every occurrence of $\bot$ by A. Therefore

$$[\![M^A]\!] = \lambda x_u^{\iota\to\iota}.\mathsf{Rec}_{\iota,(\iota\to\iota\to\iota)\to\iota} [\![M^A_{\text{base}}]\!][\![M^A_{\text{step}}]\!] n (\lambda k \lambda \ell.x_u k)$$

where

$$\begin{aligned} [\![M^A_{\text{base}}]\!] &= \lambda x_{w_0}^{\iota\to\iota\to\iota}.x_{w_0} 01 \\ [\![M^A_{\text{step}}]\!] &= \lambda n \lambda x_w^{(\iota\to\iota\to\iota)\to\iota} \lambda x_{w_1}^{\iota\to\iota\to\iota}.x_w(\lambda k \lambda \ell.x_{w_1} \ell (k+\ell)) \end{aligned}$$

Since there are no relevant formulas involved, the extracted term according to theorem 3.4.7 is

$$[\![M^A]\!](\lambda x x) = \mathsf{Rec}_{\iota,(\iota\to\iota\to\iota)\to\iota} [\![M^A_{\text{base}}]\!][\![M^A_{\text{step}}]\!] n (\lambda k \lambda \ell.k)$$

This algorithm might be easier to understand if we write it as a SCHEME program:

```
(define (fibo n)
  (fibo1 n (lambda (k l) k)))

(define (fibo1 n1 f)
  (if (= n1 0)
      (f 0 1)
      (fibo1 (- n1 1) (lambda (k l) (f l (+ k l))))))
```

E.g `(fibo 70000)` can now be easily computed.

Machine help We had to work quite a bit to obtain this program. To summarize, we had to

– give a full formalization of the classical proof,
– translate it into a corresponding constructive one, and
– extract of a program from it.

To remove some of the tedium of doing all that by hand, we certainly want machine help. In our implementation MINLOG the following interactions are necessary to produce this program.

```
(add-predicate-constant 'fib (c-arity 'nat 'nat) "fib")
(add-variable 'ff (c-arrow (c-arrow 'nat 'nat 'nat) 'nat))

(aga 'fib0 (pf "fib 0 0"))
(aga 'fib1 (pf "fib 1 1"))
(aga 'fib2 (pf "all n,k,l.fib n k -> fib(n+1)l ->
                         fib(n+2)(k+l)"))

(set-goal '? (pf "all n excl k fib n k"))
(assume 'n)
(cut (pf "all n excl k,l.fib n k ! fib(n+1)l"))

;Proof of (all n excl k,l.fib n k ! fib(n+1)l) -> excl k fib n k
(search)

;Proof of all n excl k,l.fib n k ! fib(n+1)l: Base
(ind)
(ng)
(search 1 '(fib0 1) '(fib1 1))

;and step
(ng)
(search 1 '(fib2 1))
```

This concludes construction of the classical proof. We now normalize it, transform it into a constructive one and extract a term from it.

```
(define fibonacci1 (np pproof))
(define fibonacci2
  (classical-proof-to-constr-proof fibonacci1))
(define neterm (nt (et (np fibonacci2))))
```

Here is the original printout of the extracted term `neterm` with just some indentation added.

```
(lambda (n^1)
  (((((nat-rec-at
        (quote (arrow (arrow nat (arrow nat nat)) nat)))
      (lambda (hh^2) ((hh^2 (num 0)) (num 1))))
     (lambda (n^2)
       (lambda (ff^3)
         (lambda (hh^4)
           (ff^3 (lambda (n^5)
                   (lambda (n^6)
                     ((hh^4 n^6) ((plus-nat n^5) n^6)))))))))
    n^1) (lambda (n^2) (lambda (n^3) n^2))))
```

It is rather obvious that this can be translated into the SCHEME program above.

Integer Square Roots

Let $f\colon \iota \to \iota$ be unbounded with $f0 = 0$; think of f as the square function $f(n) = n^2$. We want to prove that integer square roots always exist, or more precisely a general form of this proposition, using only the two properties of f above. We specify our problem as follows. Here f, g, n are parameters; g is used as an explicit witness of the unboundedness of f.

$$\forall n\, \neg n<f0 \to \forall n\, n<f(gn) \to (\forall m.\neg n<fm \to n<f(m+1) \to \bot) \to \bot$$

Clearly

$$\begin{array}{ll} \forall n\, \neg n<f0 & \text{is a relevant definite formula} \\ \forall n\, n<f(gn) & \text{is an irrelevant definite formula} \\ \neg n<fm & \text{is a relevant goal formula} \\ n<f(m+1) & \text{is an irrelevant goal formula,} \end{array}$$

hence theorem 3.4.7 can be applied without inserting any double negations.

To construct a derivation, assume

$$\begin{aligned} v_1 &: \forall n\, \neg n<f0, \\ v_2 &: \forall n\, n<f(gn), \\ u &: \forall m.\neg n<fm \to n<f(m+1) \to \bot. \end{aligned}$$

Our goal is $\bot$. From v_1 and u we inductively get $\forall m\, \neg n<fm$. For $m := gn$ this yields a contradiction to v_2. The derivation is

$$\dfrac{\dfrac{\dfrac{\mathsf{Ind} \qquad \neg n<f0 \qquad \forall m.\neg n<fm \to \neg n<f(m+1)}{\forall m \neg n<fm} \qquad gn}{\neg n<f(gn)} \qquad n<f(gn)}{\bot}$$

with derivation term

$$M := \lambda v_1 \lambda v_2 \lambda u.\mathsf{Ind}_{m,\neg n<fm}\, n(v_1 n)u(gn)(v_2 n),$$

Now let $A := \exists^* m.\neg n<fm \wedge n<f(m+1)$, hence

$$\begin{aligned} M^A = &\lambda v_1^{\forall n.n<f0\to A} \lambda v_2^{\forall n\, n<f(gn)} \lambda u^{\forall m.(n<fm\to A)\to n<f(m+1)\to A}. \\ &\mathsf{Ind}_{m,n<fm\to A}\, n(v_1 n)u(gn)(v_2 n) \end{aligned}$$

and therefore

$$[\![M^A]\!] =_\beta \lambda x_{v_1}^{\iota\to\iota} \lambda x_u^{\iota\to\iota\to\iota}.\mathsf{Rec}_\iota (x_{v_1} n) x_u (gn)$$

Now by (3.10) we generally have

$$D \to [\![N_D^A]\!]\ \mathbf{mr}\ D^A$$

for every relevant definite formula D. In our case, with $D = \forall n.\neg n<f0$, we can derive directly

$$(\forall n.n<f0 \to \bot) \to (\lambda n 0)\ \mathbf{mr}\ \forall n.n<f0 \to A,$$

since we can use ex-falso. So we may assume $[\![N_D^A]\!] = \lambda n 0$. Also, by (3.14) we generally have

$$x\ \mathbf{mr}\ G^A \to \neg G \to [\![H_G^A]\!](x)\ \mathbf{mr}\ A$$

for every relevant goal formula G. In our case, with $G = \neg n<fm$, we can derive directly

$$(x\ \mathbf{mr}\ n<fm \to A) \to \neg\neg n<fm \to x\ \mathbf{mr}\ A.$$

So we may assume $[\![H_G^A]\!] = \lambda x x$. Now let

$$\begin{aligned} t :&= \lambda m \lambda x.\mathbf{if}\ \neg G_1\ \mathbf{then}\ [\![H_{G_1}^A]\!](x)\ \mathbf{else}\ m \\ &= \lambda m \lambda x.\mathbf{if}\ n{<}fm\ \mathbf{then}\ x\ \mathbf{else}\ m. \end{aligned}$$

Then the extracted term according to theorem 3.4.7 is

$$\begin{aligned} [\![M^A]\!][\![N_D^A]\!]t &= \Big(\lambda x_{v_1}^{\iota\to\iota}\lambda x_u^{\iota\to\iota\to\iota}.\mathsf{Rec}_\iota(x_{v_1}n)x_u(gn)\Big)(\lambda n0)t \\ &=_\beta \mathsf{Rec}_\iota 0t(gn) \\ &= \mathsf{Rec}_\iota 0(\lambda m\lambda x.\mathbf{if}\ n{<}fm\ \mathbf{then}\ x\ \mathbf{else}\ m)(gn). \end{aligned}$$

Informally, the algorithm can be written in the form $h(gn)$ where $h\colon \iota \to \iota$ is such that

$$\begin{aligned} h(0) &= 0, \\ h(m+1) &= \begin{cases} h(m) & \text{if } n{<}f(m) \\ m & \text{else} \end{cases} \end{aligned}$$

Greatest Common Divisor

Yiannis Moschovakis suggested the following example of a classical existence proof with a quantifier-free kernel which does not obviously contain an algorithm: the gcd of two natural numbers a_1 and a_2 is a linear combination of the two. Here we treat that example as a case study for program extraction from classical proofs. In the original treatment in [5] we identified one of the predicate symbols as critical (the divisibility relation $\cdot|\cdot$). Here we show how this example can be treated in the present form of the theory, with definite and goal formulas. It turns out that the algorithm does not change; however, the method of how to extract a program from this classical proof becomes more perspicious.

Let $a, b, c, i, j, k, \ell, m, n, q, r$ denote natural numbers. Our language is determined by the constants $0, 1, +, *$, function symbols for the quotient and the remainder denoted by $q(a, c)$ and $r(a, c)$, a 4-ary function denoted by $\mathsf{abs}(k_1a_1 - k_2a_2)$ whose intended meaning is clear from the notation and an auxiliary 5-ary function f which will be defined later. We will express the intended meaning of these function symbols by stating some properties (lemmata) $v_1, \dots, v_6$ of them; these will be formulated as we need them.

Theorem 3.5.1. *For natural numbers a_1, a_2 with $0 < a_2$*

$$\exists k_1, k_2.\mathsf{abs}(k_1a_1 - k_2a_2)|a_1 \wedge \mathsf{abs}(k_1a_1 - k_2a_2)|a_2 \wedge 0 < \mathsf{abs}(k_1a_1 - k_2a_2).$$

Proof. Let a_1, a_2 be given and assume $0 < a_2$. The ideal (a_1, a_2) generated from a_1, a_2 has a least positive element c, since $0 < a_2$. This element has a representation $c = \mathsf{abs}(k_1a_1 - k_2a_2)$ with $k_1, k_2 \in \mathbb{N}$. It is a common divisor of a_1 and a_2 since otherwise the remainder $r(a_i, c)$ would be a smaller positive element of the ideal.

The number $c \in (a_1, a_2)$ dividing a_1 and a_2 is the greatest common divisor since any common divisor of a_1 and a_2 must also be a divisor of c.

The minimum principle and course-of-values induction. In order to formally write out the proof above we need to make explicit the instance of the induction scheme used implicitly in the minimum principle. The minimum principle w.r.t. a measure μ says

$$\exists \boldsymbol{k} R(\boldsymbol{k}) \to \exists \boldsymbol{k}.R(\boldsymbol{k}) \wedge \forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k}) \to R(\boldsymbol{\ell}) \to \bot$$

(in our example $R(k_1, k_2) \equiv 0 < \mathsf{abs}(k_1a_1 - k_2a_2)$ and $\mu(k_1, k_2) \equiv \mathsf{abs}(k_1a_1 - k_2a_2)$). In order to reduce this to the induction scheme we use the fact that the formula above is classically equivalent to

$$(\forall \boldsymbol{k}.R(\boldsymbol{k}) \to [\forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k}) \to R(\boldsymbol{\ell}) \to \bot] \to \bot) \to \forall \boldsymbol{k}.R(\boldsymbol{k}) \to \bot,$$

i.e. the principle of *course-of-values induction* for $R(\boldsymbol{k}) \to \bot$ w.r.t. the measure $\mu(\boldsymbol{k})$. We can write this as

$$\mathsf{Prog} \to \forall \boldsymbol{k}. R(\boldsymbol{k}) \to \bot,$$

where

$$\mathsf{Prog} := \forall \boldsymbol{k}.[\forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k}) \to R(\boldsymbol{\ell}) \to \bot] \to R(\boldsymbol{k}) \to \bot.$$

In the formal treatment of our example it will be more convenient to use the minimum principle in the form of course-of-values induction.

To prove course-of-values induction assume $w_1 \colon \mathsf{Prog}$ and prove $\forall \boldsymbol{k}. R(\boldsymbol{k}) \to \bot$. This is achieved by proving $\forall n B$, where

$$B := \forall \boldsymbol{k}.\mu(\boldsymbol{k}) < n \to R(\boldsymbol{k}) \to \bot,$$

and using B with $n := \mu(\boldsymbol{k}) + 1$. We prove $\forall n B$ by (zero-successor) induction.

Base. $B[n := 0]$ follows easily from the lemma

$$v_1 \colon \forall m. m < 0 \to \bot.$$

Step. Let n be given and assume $w_2 \colon B$. To show $B[n := n+1]$ let $\boldsymbol{k}$ be given and assume $w_3 \colon \mu(\boldsymbol{k}) < n+1$. We will derive $R(\boldsymbol{k}) \to \bot$ by using $w_1 \colon \mathsf{Prog}$ at $\boldsymbol{k}$. Hence we have to prove

$$\forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k}) \to R(\boldsymbol{\ell}) \to \bot.$$

So, let $\boldsymbol{\ell}$ be given and assume further $w_4 \colon \mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k})$. From w_4 and $w_3 \colon \mu(\boldsymbol{k}) < n+1$ we infer $\mu(\boldsymbol{\ell}) < n$ (using an arithmetical lemma). Hence, by induction hypothesis $w_2 \colon B$ at $\boldsymbol{\ell}$ we get $R(\boldsymbol{\ell}) \to \bot$.

Detailed proof of the theorem Now we repeat the proof of the theorem in some more detail using course-of-values induction. As always in classical logic, we may view the proof as an indirect one, deriving a contradiction from the assumption that the claim is false. So let a_1, a_2 be given and assume $v_0 \colon 0 < a_2$ and

$$\begin{aligned} u \colon \forall k_1, k_2. &\mathsf{abs}(k_1a_1 - k_2a_2) | a_1 \to \mathsf{abs}(k_1a_1 - k_2a_2) | a_2 \to \\ &0 < \mathsf{abs}(k_1a_1 - k_2a_2) \to \bot. \end{aligned}$$

We have to prove $\bot$ which will be achieved by proving $\forall k_1, k_2. 0 < \mathsf{abs}(k_1a_1 - k_2a_2) \to \bot$ by course-of-values induction and then specializing this formula to $k_1, k_2 = 0, 1$ and using the assumption $v_0 \colon 0 < a_2$ $(= \mathsf{abs}(0a_1 - 1a_2))$.

The principle of course-of-values induction is used with

$$N(k_1, k_2) := 0 < \mathsf{abs}(k_1a_1 - k_2a_2) \to \bot \quad \text{and} \quad \mu(k_1, k_2) := \mathsf{abs}(k_1a_1 - k_2a_2).$$

We have to show that N is progressive. To this end let k_1, k_2 be given and assume

$$u_1 \colon \forall \ell_1, \ell_2. \mu(\ell_1, \ell_2) < \mu(k_1, k_2) \to N(\ell_1, \ell_2).$$

We have to prove $N(k_1, k_2)$. So, assume $u_2 \colon 0 < \mu(k_1, k_2)$. We have to show $\bot$. This will be achieved by using the (false) assumption u at k_1, k_2. We have to prove $\mu(k_1, k_2) | a_1$ and $\mu(k_1, k_2) | a_2$. Informally, one would argue "if, say, $\mu(k_1, k_2) \not| a_1$ then the remainder $r_1 := r(a_1, \mu(k_1, k_2))$ is positive and less than $\mu(k_1, k_2)$. Furthermore we can find ℓ_1, ℓ_2 such that $r_1 = \mu(\ell_1, \ell_2)$. Altogether this contradicts the assumption u_1". More formally, to prove $\mu(k_1, k_2) | a_1$ we use the lemma

$$v_2 \colon \forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to c | a$$

at a_1, $q_1 := q(a_1, \mu(k_1, k_2))$ (the quotient), $\mu(k_1, k_2)$ and $r_1 := r(a_1, \mu(k_1, k_2))$ (the remainder). We have to prove the premises

$$a_1 = q_1\mu(k_1, k_2) + r_1 \quad \text{and} \quad 0 < r_1 \to \bot$$

of the instantiated lemma v_2. Here we need the lemmata

$$\begin{aligned} &v_3 \colon \forall a, c. 0 < c \to a = q(a,c)c + r(a,c), \\ &v_4 \colon \forall a, c. 0 < c \to r(a,c) < c \end{aligned}$$

specifying the functions quotient and remainder. Now the first premise follows immediately from lemma v_3 and $u_2 \colon 0 < \mu(k_1, k_2)$. To prove the second premise, $0 < r_1 \to \bot$, we assume $u_3 \colon 0 < r_1$ and show $\bot$. First we compute ℓ_1, ℓ_2 such that $r_1 = \mu(\ell_1, \ell_2)$. This is done by some auxiliary function f, defined by

$$f(a_1, a_2, k_1, k_2, q) := \begin{cases} qk_1 - 1, & \text{if } k_2 a_2 < k_1 a_1 \text{ and } 0 < q; \\ qk_1 + 1, & \text{otherwise.} \end{cases}$$

f satisfies the lemma

$$v_5 \colon \forall a_1, a_2, k_1, k_2, q, r. a_1 = q \cdot \mu(k_1, k_2) + r \to r = \mu(f(a_1, a_2, k_1, k_2, q), qk_2).$$

Hence we let $\ell_1 := f(a_1, a_2, k_1, k_2, q_1)$ and $\ell_2 := q_1 k_2$. Now we have $\mu(\ell_1, \ell_2) = r_1 < \mu(k_1, k_2)$ by v_5, u_2 and v_4, as well as $0 < r_1 = \mu(\ell_1, \ell_2)$ by u_3 and v_5. Therefore, we get $\bot$ by u_1 at ℓ_1, ℓ_2 (using some equality lemmata). This completes the proof of $\mu(k_1, k_2) | a_1$. $\mu(k_1, k_2) | a_2$ is proved similarly using the lemma

$$v_6 \colon \forall a_1, a_2, k_1, k_2, q, r. a_2 = q \cdot \mu(k_1, k_2) + r \to r = \mu(qk_1, f(a_2, a_1, k_2, k_1, q)).$$

Turning the lemmata and the claim into definite and goal formulas In order to make our general theory applicable to the present example we have to make sure that the lemmata assert definite formulas and the claim is given by goal formulas. This is almost the case already, with the sole exeption of

$$v_2 \colon \forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to c | a,$$

which is not definite (we have concluded something positive, i.e. $c|a$, from something negative, i.e. $0 < r \to \bot$). We now make it definite by inserting a double negation in front of the final conclusion $c|a$, i.e. we replace the formula of v_2 by

$$\forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to (c | a \to \bot) \to \bot.$$

Clearly this will not affect our proof in any essential way; however, it will make a difference when it comes to extract the program.

Formal proof The proof of the principle of course-of-values induction and the proof of the theorem were given in such a detail that it is now easy to formalize them completely. Only some arguments concerning $<$ and $=$ were left implicit, but since these will be irrelevant formulas we don't need to worry about these omissions.

$$\begin{aligned} M :=\, &\lambda v_0^{0<a_2} \\ &\lambda v_1^{\forall m. m<0\to\bot} \\ &\lambda v_2^{\forall a,q,c,r. a=qc+r\to(0<r\to\bot)\to(c|a\to\bot)\to\bot} \\ &\lambda v_3^{\forall a,c. 0<c\to a=q(a,c)c+r(a,c)} \\ &\lambda v_4^{\forall a,c. 0<c\to r(a,c)<c} \\ &\lambda v_5^{\forall a_1,a_2,k_1,k_2,q,r. a_1=q\cdot\mu(k_1,k_2)+r\to r=\mu(f(a_1,a_2,k_1,k_2,q),qk_2)} \\ &\lambda v_6^{\forall a_1,a_2,k_1,k_2,q,r. a_2=q\cdot\mu(k_1,k_2)+r\to r=\mu(qk_1,f(a_2,a_1,k_2,k_1,q))} \\ &\lambda u^{\forall \boldsymbol{k}. \mu(\boldsymbol{k})|a_1\to\mu(\boldsymbol{k})|a_2\to 0<\mu(\boldsymbol{k})\to\bot}. \\ &M_{\mathrm{cvind}}^{\mathsf{Prog}\to\forall \boldsymbol{k}. 0<\mu(\boldsymbol{k})\to\bot} M_{\mathrm{prog}} 0 1 (L^{0<\mu(0,1)}[v_0]) \end{aligned}$$

where

$$
\begin{aligned}
M_{\mathrm{cvind}} &= \lambda w_1^{\mathsf{Prog}} \lambda \boldsymbol{k}.\mathsf{Ind}_{n,B} M_{\mathrm{base}} M_{\mathrm{step}}(\mu(\boldsymbol{k})+1)\boldsymbol{k} L^{\mu(\boldsymbol{k})<\mu(\boldsymbol{k})+1}, \\
M_{\mathrm{base}} &= \lambda \boldsymbol{k} \lambda w_0^{\mu(\boldsymbol{k})<0} \lambda \tilde{w}_0{}^{0<\mu(\boldsymbol{k})}.v_1 \mu(\boldsymbol{k}) w_0, \\
M_{\mathrm{step}} &= \lambda n \lambda w_2^B \lambda \boldsymbol{k} \lambda w_3^{\mu(\boldsymbol{k})<n+1}.w_1 \boldsymbol{k}(\lambda \boldsymbol{\ell} \lambda w_4^{\mu(\boldsymbol{\ell})<\mu(\boldsymbol{k})}.w_2 \boldsymbol{\ell}(L^{\mu(\boldsymbol{\ell})<n}[w_4, w_3])),
\end{aligned}
$$

$$
\begin{aligned}
M_{\mathrm{prog}} &= \lambda \boldsymbol{k} \lambda u_1^{\forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell})<\mu(\boldsymbol{k}) \to 0<\mu(\boldsymbol{\ell}) \to \bot} \lambda u_2^{0<\mu(\boldsymbol{k})}. \\
&\qquad M_{\mathrm{div}_2}^{(\mu(\boldsymbol{k})|a_2 \to \bot) \to \bot} \lambda u_3^{\mu(\boldsymbol{k})|a_2}.M_{\mathrm{div}_1}^{(\mu(\boldsymbol{k})|a_1 \to \bot) \to \bot} \lambda u_4^{\mu(\boldsymbol{k})|a_1}.u \boldsymbol{k} u_4 u_3 u_2, \\
M_{\mathrm{div}_i} &= v_2 a_i q_i(\boldsymbol{k}) \mu(\boldsymbol{k}) r_i(\boldsymbol{k})(L^{a_i = q_i(\boldsymbol{k})\mu(\boldsymbol{k}) + r_i(\boldsymbol{k})}[v_3, u_2]) M_{\not<_i}^{0<r_i(\boldsymbol{k}) \to \bot}, \\
M_{\not<_i} &= \lambda u_{3,i}^{0<r_i(\boldsymbol{k})}.u_1 \boldsymbol{\ell}_i(\boldsymbol{k})(L^{\mu(\boldsymbol{\ell}_i(\boldsymbol{k}))<\mu(\boldsymbol{k})}[v_{4+i}, v_3, u_2, v_4])(L^{0<\mu(\boldsymbol{\ell}_i(\boldsymbol{k}))}[u_{3,i}, v_{4+i}, v_3, u_2]).
\end{aligned}
$$

Here we have used the abbreviations

$$
\begin{aligned}
\mathsf{Prog} &= \forall \boldsymbol{k}.[\forall \boldsymbol{\ell}.\mu(\boldsymbol{\ell}) < \mu(\boldsymbol{k}) \to 0 < \mu(\boldsymbol{\ell}) \to \bot] \to 0 < \mu(\boldsymbol{k}) \to \bot \\
B &= \forall \boldsymbol{k}.\mu(\boldsymbol{k}) < n \to 0 < \mu(\boldsymbol{k}) \to \bot
\end{aligned}
$$

Term extraction Let

$$A := \exists^* \boldsymbol{k}.\mu(\boldsymbol{k})|a_1 \wedge \mu(\boldsymbol{k})|a_2 \wedge 0 < \mu(\boldsymbol{k}),$$

so $\tau(A) = \boldsymbol{\iota} := \iota \times \iota$. Let $C^A := C[\bot := A]$, hence

$$
\begin{aligned}
&\tau(\mathsf{Prog}^A) = \boldsymbol{\iota} \to (\boldsymbol{\iota} \to \boldsymbol{\iota}) \to \boldsymbol{\iota}, \\
&\tau(B^A) = \boldsymbol{\iota} \to \boldsymbol{\iota}.
\end{aligned}
$$

Furthermore recall $[\![\mathsf{Ind}_{n,B^A}]\!] = \mathsf{Rec}_{\iota,\boldsymbol{\iota}\to\boldsymbol{\iota}}$ where

$$\mathsf{Rec}_{\iota,\boldsymbol{\iota}\to\boldsymbol{\iota}} \colon (\boldsymbol{\iota} \to \boldsymbol{\iota}) \to (\iota \to (\boldsymbol{\iota} \to \boldsymbol{\iota}) \to (\boldsymbol{\iota} \to \boldsymbol{\iota})) \to \iota \to (\boldsymbol{\iota} \to \boldsymbol{\iota})$$

Note that $\mathsf{Rec}_{\iota,\boldsymbol{\iota}\to\boldsymbol{\iota}}$ may be viewed as simultaneous primitive recursion operators $\mathbf{Rec} = (\mathsf{Rec}_1, \mathsf{Rec}_2)$ with

$$
\begin{aligned}
&\mathsf{Rec}_i \boldsymbol{y} \boldsymbol{f} 0 = y_i, \\
&\mathsf{Rec}_i \boldsymbol{y} \boldsymbol{f}(z+1) = f_i z(\mathsf{Rec}_1 \boldsymbol{y} \boldsymbol{f} z)(\mathsf{Rec}_2 \boldsymbol{y} \boldsymbol{f} z).
\end{aligned}
$$

Now we are prepared to compute the extracted term. Let M^A denote the result of replacing every formula C in the derivation M by C^A (note that M^A generally does *not* derive A).

$$[\![M^A]\!] = \lambda x_{v_1}^{\iota \to \boldsymbol{\iota}} \lambda x_{v_2}^{\iota^4 \to \boldsymbol{\iota} \to \boldsymbol{\iota} \to \boldsymbol{\iota}} \lambda x_u^{\boldsymbol{\iota} \to \boldsymbol{\iota}}.[\![M_{\mathrm{cvind}}^A]\!][\![M_{\mathrm{prog}}^A]\!]01$$

where

$$
\begin{aligned}
[\![M_{\mathrm{cvind}}^A]\!] &= \lambda x_{w_1}^{\boldsymbol{\iota} \to (\boldsymbol{\iota} \to \boldsymbol{\iota}) \to \boldsymbol{\iota}} \lambda \boldsymbol{k}.\mathsf{Rec}[\![M_{\mathrm{base}}^A]\!][\![M_{\mathrm{step}}^A]\!](\mu(\boldsymbol{k})+1)\boldsymbol{k} \\
[\![M_{\mathrm{base}}^A]\!] &= \lambda \boldsymbol{k}.x_{v_1} \mu(\boldsymbol{k}) \\
[\![M_{\mathrm{step}}^A]\!] &= \lambda n \lambda x_{w_2}^{\boldsymbol{\iota} \to \boldsymbol{\iota}} \lambda \boldsymbol{k}.x_{w_1} \boldsymbol{k}(\lambda \boldsymbol{\ell}.x_{w_2} \boldsymbol{\ell}),
\end{aligned}
$$

$$
\begin{aligned}
[\![M_{\mathrm{prog}}^A]\!] &= \lambda \boldsymbol{k} \lambda x_{u_1}^{\boldsymbol{\iota} \to \boldsymbol{\iota}}.[\![M_{\mathrm{div}_2}^A]\!]([\![M_{\mathrm{div}_1}^A]\!](x_u \boldsymbol{k})) \\
[\![M_{\mathrm{div}_i}^A]\!] &= x_{v_2} a_i q_i(\boldsymbol{k}) \mu(\boldsymbol{k}) r_i(\boldsymbol{k})[\![M_{\not<_i}^A]\!] \\
[\![M_{\not<_i}^A]\!] &= x_{u_1} \boldsymbol{\ell}_i(\boldsymbol{k}).
\end{aligned}
$$

Now by (3.10) we generally have $D \to [\![N_D^A]\!]$ $\mathbf{mr}$ D^A for every relevant definite formula D. In our case for $D_1 = \forall m.m < 0 \to \bot$ we can derive directly

$$(\forall m.m < 0 \to \bot) \to (\lambda n 0) \ \mathbf{mr} \ \forall m.m < 0 \to A,$$

since we can use ex-falso. So we may assume $[\![N_{D_1}^A]\!] = \lambda n 0$. Similarly for

$$D_2 = \forall a, q, c, r. a = qc + r \to (0 < r \to \bot) \to (c|a \to \bot) \to \bot \quad \text{we have}$$
$$D_2^A = \forall a, q, c, r. a = qc + r \to (0 < r \to A) \to (c|a \to A) \to A.$$

Now an informal proof of $D_2 \to D_2^A$ runs as follows. Assume the hypotheses. We prove A by cases on $0 < r$. In case $0 < r$ the hypothesis $0 < r \to A$ yields A. In case $0 < r \to \bot$ we obtain $(c|a \to \bot) \to \bot$ and hence $c|a$. Now the hypothesis $c|a \to A$ yields A.

From this proof we can derive directly

$$D_2 \to (\lambda a \lambda q \lambda c \lambda r \lambda \boldsymbol{x} \lambda \boldsymbol{y}.\textbf{if } 0 < r \textbf{ then } \boldsymbol{x} \textbf{ else } \boldsymbol{y}) \textbf{ mr } D_2^A.$$

So we may assume $[\![N_{D_2}^A]\!] = \lambda a \lambda q \lambda c \lambda r \lambda \boldsymbol{x} \lambda \boldsymbol{y}.\textbf{if } 0 < r \textbf{ then } \boldsymbol{x} \textbf{ else } \boldsymbol{y}$.

Moreover, there is no relevant goal formula, hence $t = \lambda \boldsymbol{y}\boldsymbol{y}$. Then the extracted term according to theorem 3.4.7 is

$$[\![M^A]\!][\![N_{D_1}^A]\!][\![N_{D_2}^A]\!]t =_\beta [\![M_{\text{cvind}}^A]\!]'[\![M_{\text{prog}}^A]\!]'01$$

where $'$ indicates substitution of $[\![N_{D_1}^A]\!]$, $[\![N_{D_2}^A]\!]$, t for x_{v_1}, x_{v_2}, x_u, so

$$\begin{aligned}
[\![M_{\text{cvind}}^A]\!]' &=_{\beta\eta} \lambda x_{w_1}^{\iota \to (\iota \to \iota) \to \iota} \lambda \boldsymbol{k}'.\mathsf{Rec}(\lambda \boldsymbol{k}.\boldsymbol{0})(\lambda n \lambda x_{w_2} \lambda \boldsymbol{k}.x_{w_1} \boldsymbol{k} x_{w_2})(\mu(\boldsymbol{k}') + 1)\boldsymbol{k}', \\
[\![M_{\text{prog}}^A]\!]' &=_\beta \lambda \boldsymbol{k} \lambda x_{u_1}^{\iota \to \iota}.[\![M_{\text{div}_2}^A]\!]'([\![M_{\text{div}_1}^A]\!]'\boldsymbol{k}), \\
[\![M_{\text{div}_i}^A]\!]' &=_\beta (\lambda \boldsymbol{x} \lambda \boldsymbol{y}.\textbf{if } 0 < \boldsymbol{r}_i(\boldsymbol{k}) \textbf{ then } \boldsymbol{x} \textbf{ else } \boldsymbol{y})(x_{u_1} \boldsymbol{\ell}_i(\boldsymbol{k}))
\end{aligned}$$

and hence

$$\begin{aligned}
[\![M_{\text{prog}}^A]\!]' =_\beta \lambda \boldsymbol{k} \lambda x_{u_1}.&\textbf{if } 0 < r_2(\boldsymbol{k}) \textbf{ then } x_{u_1} \boldsymbol{\ell}_2(\boldsymbol{k}) \textbf{ else} \\
&(\textbf{if } 0 < r_1(\boldsymbol{k}) \textbf{ then } x_{u_1} \boldsymbol{\ell}_1(\boldsymbol{k}) \textbf{ else } \boldsymbol{k})
\end{aligned}$$

Therefore we obtain as extracted algorithm, using the fact that $\mu(0,1) = a_2$,

$$\begin{aligned}
[\![M^A]\!][\![N_{D_1}^A]\!][\![N_{D_2}^A]\!]t =_\beta \mathsf{Rec}&(\lambda \boldsymbol{k}.\boldsymbol{0}) \\
&(\lambda n \lambda x_{w_2}^{\iota \to \iota} \lambda \boldsymbol{k}.\textbf{if } 0 < r_2(\boldsymbol{k}) \textbf{ then } x_{w_2} \boldsymbol{\ell}_2(\boldsymbol{k}) \textbf{ else} \\
&\qquad (\textbf{if } 0 < r_1(\boldsymbol{k}) \textbf{ then } x_{w_2} \boldsymbol{\ell}_1(\boldsymbol{k}) \textbf{ else } \boldsymbol{k})) \\
&(a_2 + 1)01
\end{aligned}$$

To make this algorithm more readable we may write $[\![M^A]\!][\![N_{D_1}^A]\!][\![N_{D_2}^A]\!]t = h(a_2 + 1, 0, 1)$, where

$$\begin{aligned}
h(0, \boldsymbol{k}) &:= \boldsymbol{0}, \\
h(n+1, \boldsymbol{k}) &:= \textbf{if } 0 < r_2(\boldsymbol{k}) \textbf{ then } h(n, \boldsymbol{\ell}_2(\boldsymbol{k})) \textbf{ else} \\
&\qquad (\textbf{if } 0 < r_1(\boldsymbol{k}) \textbf{ then } h(n, \boldsymbol{\ell}_1(\boldsymbol{k})) \textbf{ else } \boldsymbol{k})
\end{aligned}$$

Example Let us try out the extracted algorithm to compute coefficients k_1, k_2 such that $\gcd(66, 27) = |k_1 \cdot 66 - k_2 \cdot 27|$.

$$\begin{array}{ll}
h(28, 0, 1) & \mu(0,1) = 27 \\
& 0 < r_1 = 12 \quad q_1 = 2 \\
& \underbrace{q_1 k_1}_{0} \pm 1, \quad \underbrace{q_1 k_2}_{2} \qquad -1, \text{ if } k_2 a_2 < \underbrace{k_1 a_1}_{0} \quad \text{No} \\
& 1, \quad 2 \\
h(27, 1, 2) & \mu(1,2) = 12 \\
& 0 < r_1 = 6 \quad q_1 = 5 \\
& \underbrace{q_1 k_1}_{5 \cdot 1} \pm 1, \quad \underbrace{q_1 k_2}_{5 \cdot 2} \qquad -1, \text{ if } \underbrace{k_2 a_2}_{2 \cdot 27} < \underbrace{k_1 a_1}_{1 \cdot 66} \quad \text{Yes} \\
& 4, \quad 10
\end{array}$$

$$
\begin{array}{ll}
h(26,4,10) & \mu(4,10) = |4\cdot 66 - 10\cdot 27| = |264-270| = 6 \\
& 6|66 \\
& 0 < r_2 = 3 \quad q_2 = 4 \\
& \underbrace{q_2k_1,}_{4\cdot 4} \quad \underbrace{q_2k_2}_{4\cdot 10}\pm 1 \qquad -1,\ \text{if}\ \ \underbrace{k_1a_1}_{4\cdot 66=264} < \underbrace{k_2a_2}_{10\cdot 27=270} \qquad \text{Yes} \\
& 16, \quad 39 \\
h(25,16,39) & \mu(16,39) = |16\cdot 66 - 39\cdot 27| = |1056-1053| = 3 \\
& 3|66 \\
& 3|27 \\
& \text{Result: } 16, 39
\end{array}
$$

Note that, although $3 = |16\cdot 66 - 39\cdot 27|$ is the least positive element of the ideal $(66, 27)$, the coefficients 16, 39 are not minimal. The minimal coefficients are 2, 5.

Remarks 1. As one sees from this example the recursion parameter n is not really used in the computation but just serves as a counter or more precisely as an upper bound for the number of steps until both remainders are zero. This will always happen if the induction principle is used only in the form of the minimum principle (or, equivalently, course-of-values induction). Because then in the extracted terms of course-of-values induction, the step $[\![M^A_{\text{step}}]\!] = \lambda n \lambda x^{\iota\to\iota}_{w_2} \lambda \boldsymbol{k}.x_{w_1}\boldsymbol{k}(\lambda \boldsymbol{\ell}.x_{w_2}\boldsymbol{\ell})$ has in its kernel no free occurrence of n.

2. If one removes n according to the previous remark it becomes clear that our gcd algorithm is similar to Euclid's. The only difference lies in the fact that we have kept a_1, a_2 fixed in our proof whereas Euclid changes a_1 to a_2 and a_2 to $r(a_1, a_2)$ provided $r(a_1, a_2) > 0$ (using the fact that this doesn't change the ideal).

3. There is an interesting phenomenon which may occur if we extract a program from a classical proof which uses the minimum principle. Consider as a simple example the wellfoundedness of $<$,

$$\forall g^{\iota\to\iota} \exists k. g(k+1) < g(k) \to \bot.$$

If one formalizes the classical proof "choose k such that $g(k)$ is minimal" and extracts a program one might expect that it computes a k such that $g(k)$ is minimal. But this is impossible! In fact the program computes the least k such that $g(k+1) < g(k) \to \bot$ instead. This discrepancy between the classical proof and the extracted program didn't show up in our gcd example since there was only one $c = \mu(k) > 0$ such that c divides a_1 and a_2, whereas in the example above there may be different $c = g(k)$ such that $g(k+1) < g(k) \to \bot$.

3.6 Notes

Proofs of termination for terms of Gödel's system T can be found in many places in the literature. Troelstra in [11], p. 107–108 gives a proof which either uses König's Lemma (or the fan theorem) or else the Church-Rosser Theorem for $\longrightarrow$. The short proof given in section 3.1 is due to Holger Benl [2, p.8], Ulrich Berger and Ralph Matthes; it works for arbitrary arithmetical terms and does not use the Church-Rosser Theorem. A similar approach has been used in a more general type theoretic context by Thorsten Altenkirch in his thesis [1].

A proof which works for closed terms only (but in a general setting where bounded fixed point operators are allowed) is given in [26]. A simple proof of uniqueness of normal forms, which uses the technique of parallel reduction (originating from W.W. Tait) has been given by Takahashi in [28].

The treatment in section 3.4 of extracting computational content from classical proofs is based on work of Berger and Schwichtenberg [4, 5]. There a set L of formulas steering the translation had to be considered, and from L one could read off the set $\mathcal{C}_L$ of L-critical predicate symbols. The refined A-translation then consists in replacing every L-critial atom by its double negation. The refinement we present in section 3.4 is due to Buchholz [9]; it consists in defining the slightly more general notions of *definite* and *goal* formulas. Compared with the previous approach, we now – by inserting double negations – have to make sure that all assumption formulas (lemmas) are definite and the kernel of the formula to be proved is a goal formula. This is slightly more flexible than before, since the same predicate symbol can be trated differently at each of its occurrences.

References

1. Thorsten Altenkirch. *Constructions, Inductive Types and Strong Normalization.* PhD thesis, The University of Edinburgh, Department of Computer Science, Edinburgh, 1993.
2. Holger Benl. Konstruktive Interpretation induktiver Definitionen. Master's thesis, Mathematisches Institut der Universität München, 1998.
3. Holger Benl and Helmut Schwichtenberg. Formal correctness proofs of functional programs: Dijkstra's algorithm, a case study. In U. Berger and H. Schwichtenberg, editors, *Computational Logic*, volume 165 of *Series F: Computer and Systems Sciences*, pages 113–126. Proceedings of the NATO Advanced Study Institute on Computational Logic, held in Marktoberdorf, Germany, July 29 – August 10, 1997, Springer Verlag, Berlin, Heidelberg, New York, 1999.
4. Ulrich Berger and Helmut Schwichtenberg. Program extraction from classical proofs. In D. Leivant, editor, *Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, IN, USA, October 1994*, volume 960 of *Lecture Notes in Computer Science*, pages 77–97. Springer Verlag, Berlin, Heidelberg, New York, 1995.
5. Ulrich Berger and Helmut Schwichtenberg. The greatest common divisor: a case study for program extraction from classical proofs. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs. International Workshop TYPES '95, Torino, Italy, June 1995. Selected Papers*, volume 1158 of *Lecture Notes in Computer Science*, pages 36–46. Springer Verlag, Berlin, Heidelberg, New York, 1996.
6. Ulrich Berger, Helmut Schwichtenberg, and Monika Seisenberger. The Warshall Algorithm and Dickson's Lemma: Two Examples of Realistic Program Extraction. In preparation, 1999.
7. E.W. Beth. Semantic construction of intuitionistic logic. *Mededelingen de KNAW N.S.*, 19(11), 1956.
8. E.W. Beth. *The foundations of mathematics.* North–Holland, Amsterdam, 1959.
9. Wilfried Buchholz. Zum Seminar über Programmextraktion. Notes, 1997.
10. Roy Dyckhoff. Contraction–free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, 57:793–807, 1992.
11. Anne S. Troelstra (editor). *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, volume 344 of *Lecture Notes in Mathematics.* Springer Verlag, Berlin, Heidelberg, New York, 1973.
12. Melvin Fitting. *Intuitionistic Logic, Model Theory and Forcing.* North–Holland, Amsterdam, 1969.
13. Harvey Friedman. Equality between functionals. In R. Parikh, editor, *Logic Colloquium, Lecture Notes in Mathematics 453*, pages 22–37. Springer, 1975.
14. Harvey Friedman. Classically and intuitionistically provably recursive functions. In D.S. Scott and G.H. Müller, editors, *Higher Set Theory*, volume 669 of *Lecture Notes in Mathematics*, pages 21–28. Springer Verlag, Berlin, Heidelberg, New York, 1978.
15. Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934.
16. Gerhard Gentzen. Beweisbarkeit und Unbeweisbarkeit von Anfangsfällen der transfiniten Induktion in der reinen Zahlentheorie. *Mathematische Annalen*, 119:140–161, 1943.
17. V.I. Glivenko. Sur quelques points de la logique de M. Brouwer. *Bull. Soc. Math. Belg.*, 15:183–188, 1929.
18. Jörg Hudelmaier. A PROLOG Program for Intuitionistic Logic. SNS–Bericht 88–28, Seminar für natürlich–sprachliche Systeme der Universität Tübingen, 1988.
19. Jörg Hudelmaier. An $o(n \log n)$–space decision procedure for intuitionistic propositional logic. *J. Logic Computation*, 3(1):63–75, 1993.
20. I. Johannson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 4:119–136, 1937.
21. Saul A. Kripke. Semantical analysis of intuitionistic logic I. In J. Crossley and M. Dummett, editors, *Formal Systems and Recursive Functions*, pages 93–130. North–Holland, Amsterdam, 1965.
22. Sigekatu Kuroda. Intuitionistische Untersuchungen der formalistischen Logik. *Nagoya Mathematical Journal*, 2:35–47, 1951.
23. M.H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.
24. V.P. Orevkov. Lower bounds for increasing complexity of derivations after cut elimination. *Zapiski Nauchnykh Seminarov Leningradskogo*, 88:137–161, 1979.
25. U. Moscato P. Miglioli and M. Ornaghi. How to avoid duplications in refutation systems for intuitionistic logic and Kuroda logic. Rapporto interno 99-93, Universita degli Studi Milano, Dip. Sc. Informazione, 1993.

26. Helmut Schwichtenberg and Stanley S. Wainer. Ordinal bounds for programs. In P. Clote and J. Remmel, editors, *Feasible Mathematics II*, pages 387–406. Birkhäuser, Boston, 1995.
27. Richard Statman. Bounds for proof-search and speed-up in the predicate calculus. *Annals of Mathematical Logic*, 15:225–287, 1978.
28. Masako Takahashi. Parallel reductions in λ–calculus. *Information and Computation*, 118:120–127, 1995.
29. Anne S. Troelstra. *Choice Sequences.* Oxford Logic Guides. Clarendon Press, Oxford, 1977.
30. Anne S. Troelstra and Dirk van Dalen. *Constructivism in Mathematics. An Introduction*, volume 121, 123 of *Studies in Logic and the Foundations of Mathematics.* North–Holland, Amsterdam, 1988.
31. Klaus Weich. Decision procedures for intuitionistic propositional logic by program extraction. In H. de Swart, editor, *Automated Reasoning with Analytic Tableaux and Related Methods. Proceedings Tableaux '98*, LNAI, pages 292–306. Springer Verlag, Berlin, Heidelberg, New York, 1998.

Index