Mathematical Logic

Helmut Schwichtenberg

Mathematisches Institut der LMU, Wintersemester 2024

Contents

Preface

This is the script for the course "Mathematical Logic" at LMU, held in Wintersemester 2024/2025. It is mainly based on the textbook "Proofs and Computations" in the references. I would like to thank Nicolar Gruben and Konstantin Grabmann for taking care of the exercises, and the students for their active participation.

München, 20. February 2024

Helmut Schwichtenberg

CHAPTER 1

Formal Proofs

The main subject of Mathematical Logic is mathematical proof. In this introductory chapter we deal with the basics of formalizing such proofs and, via normalization, analysing their structure. The system we pick for the representation of proofs is Gentzen's natural deduction from (1935). 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 proceed anyway. Second, formal proofs in natural deduction are closely related (via the so-called Curry-Howard correspondence) to terms in typed lambda 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 lambda calculus.

An underlying theme of this chapter is to bring out the constructive content of logic, particularly in regard to the relationship between minimal and classical logic. For us the latter is most appropriately viewed as a subsystem of the former. This approach will reveal some interesting aspects of proofs, e.g., that it is possible and useful to distinguish beween existential proofs that actually construct witnessing objects, and others that don't.

As an example for a non-constructive existence proof, 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 $\sqrt{2}^{\sqrt{2}}$ is rational. Choose $a = \sqrt{2}$ and $b = \sqrt{2}$. Then a, b are irrational and by assumption a^b is rational.

Case $\sqrt{2}^{\sqrt{2}}$ is irrational. 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.

Weyl (1921) gave a somewhat drastic description of this situation:

> Ein Existentialsatz – etwa "es gibt eine gerade Zahl" – ist überhaupt kein Urteil im eigentlichen Sinne, das einen Sachverhalt behauptet; Existentialsachverhalte sind eine leere Erfindung der Logiker. "2 ist eine gerade Zahl", das ist ein wirkliches, einem Sachverhalt Ausdruck gebendes Urteil; "es gibt eine gerade Zahl" ist nur ein aus diesem Urteil gewonnenes Urteilsabstrakt. Bezeichne ich Erkenntnis als einen wertvollen Schatz, so ist das Urteilsabstrakt ein Papier, welches das Vorhandensein eines Schatzes anzeigt, ohne jedoch zu verraten, an welchem Ort. Sein einziger Wert kann darin liegen, daß es mich antreibt, nach dem Schatze zu suchen.

1.1. Natural deduction

The rules of natural deduction come in pairs: we have an introduction and an elimination rule for each of the logical connectives. The resulting system is called *minimal logic*; it was introduced by Kolmogorov (1932), Gentzen (1935) and Johansson (1937). Notice that no negation is yet present. If we go on and require *ex-falso-quodlibet* for the nullary propositional symbol $\bot$ ("falsum") we can embed *intuitionistic logic* with negation as $A \to \bot$. To embed classical logic, we need to go further and add as an axiom schema the principle of *indirect proof*, also called *stability* ($\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x})$ for relation symbols R), but then it is appropriate to restrict to the language based on $\to$, $\forall$, $\bot$ and $\wedge$. The reason for this restriction is that we can neither prove $\neg\neg\exists_x A \to \exists_x A$ nor $\neg\neg(A \vee B) \to A \vee B$, for there are countermodels to both (the former is Markov's scheme). However, we can prove them for the classical existential quantifier and disjunction defined by $\neg\forall_x\neg A$ and $\neg A \to \neg B \to \bot$. Thus we need to make a distinction between two kinds of "exists" and two kinds of "or": the classical ones are "weak" and the non-classical ones "strong" since they have constructive content. In situations where both kinds occur together we must mark the distinction, and we shall do this by writing a tilde above the weak disjunction and existence symbols thus $\tilde{\vee}, \tilde{\exists}$.

1.1.1. Terms and formulas. Let a countably 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.

(i) For every natural number $n \geq 0$ a (possible empty) set of n-ary *relation symbols* (or *predicate symbols*). 0-ary relation symbols are called *propositional symbols*. $\bot$ (read "falsum") is required as a fixed propositional symbol. The language will *not*, unless stated otherwise, contain $=$ as a primitive. Binary relation symbols can be marked as infix.

(ii) For every natural number $n \geq 0$ a (possible empty) set of n-ary *function symbols*. 0-ary function symbols are called *constants*. Binary function symbols can again be marked as infix.

We assume that all these sets of variables, relation and function symbols are disjoint. $\mathcal{L}$ is kept fixed and will only be mentioned when necessary.

Terms are inductively defined as follows.

(i) Every variable is a term.

(ii) Every constant is a term.

(iii) If $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol with $n \geq 1$, then $f(t_1, \ldots, t_n)$ is a term. (If t, s are terms and $\circ$ is a binary function symbol, then $(t \circ s)$ is a term.)

From terms one constructs *prime formulas*, also called *atomic formulas* or just *atoms*: If $t_1, \ldots, t_n$ are terms and R is an n-ary relation symbol, then $R(t_1, \ldots, t_n)$ is a prime formula. (If t, s are terms and $\sim$ is a binary relation symbol, then $(t \sim s)$ is a prime formula.)

Formulas are inductively defined from prime formulas by

(i) Every prime formula is a formula.

(ii) If A and B are formulas, then so are $(A \to B)$ ("if A then B"), $(A \wedge B)$ ("A and B") and $(A \vee B)$ ("A or B").

(iii) If A is a formula and x is a variable, then $\forall_x A$ ("A holds for all x") and $\exists_x A$ ("there is an x such that A") are formulas.

Negation is defined by

$$\neg A := (A \to \bot).$$

We shall often need to do induction on the height, denoted $|A|$, of formulas A. This is defined as follows: $|P| = 0$ for atoms P, $|A \circ B| = \max(|A|, |B|) + 1$ for binary operators $\circ$ (i.e., $\to, \wedge, \vee$) and $|\circ A| = |A| + 1$ for unary operators $\circ$ (i.e., $\forall_x$, $\exists_x$).

1.1.2. Substitution, free and bound variables. Expressions $\mathcal{E}, \mathcal{E}'$ which differ only in the names of bound (occurrences of) variables will be regarded as identical. This is sometimes expressed by saying that $\mathcal{E}$ and $\mathcal{E}'$ are α-equal. In other words, we are only interested in expressions "modulo renaming of bound variables". There are methods of finding unique representatives for such expressions, e.g., the name-free terms of de Bruijn (1972). For the human reader such representations are less convenient, so we shall stick to the use of bound variables.

In the definition of "substitution of expression $\mathcal{E}'$ for variable x in expression $\mathcal{E}$", either one requires that *no* variable free in $\mathcal{E}'$ becomes bound by a variable-binding operator in $\mathcal{E}$, when the free occurrences of x are replaced by $\mathcal{E}'$ (also expressed by saying that there must be no "clashes of variables"), "*$\mathcal{E}'$ is free for x in $\mathcal{E}$*", or the substitution operation is taken to involve a systematic renaming operation for the bound variables, avoiding clashes. Having stated that we are only interested in expressions modulo renaming bound variables, we can without loss of generality assume that substitution is always possible.

Also, it is never a real restriction to assume that distinct quantifier occurrences are followed by distinct variables, and that the sets of bound and free variables of a formula are disjoint.

NOTATION. "FV" is used for the (set of) free variables of an expression; so $\mathrm{FV}(t)$ is the set of variables free in the term t, $\mathrm{FV}(A)$ the set of variables free in formula A etc. A formula A is said to be *closed* if $\mathrm{FV}(A) = \emptyset$.

$\mathcal{E}[x := t]$ denotes the result of substituting the term t for the variable x in the expression $\mathcal{E}$. Similarly, $\mathcal{E}[\vec{x} := \vec{t}]$ is the result of *simultaneously* substituting the terms $\vec{t} = t_1, \ldots, t_n$ for the variables $\vec{x} = x_1, \ldots, x_n$, respectively.

In a given context we shall adopt the following convention. Once a formula has been introduced as $A(x)$, i.e., A with a designated variable x, we write $A(t)$ for $A[x := t]$, and similarly with more variables.

1.1.3. Subformulas. Unless stated otherwise, the notion of *subformula* will be that defined by Gentzen.

DEFINITION. (Gentzen) subformulas of A are defined by

(a) A is a subformula of A;
(b) if $B \circ C$ is a subformula of A then so are B, C, for $\circ = \rightarrow, \wedge, \vee$;
(c) if $\forall_x B(x)$ or $\exists_x B(x)$ is a subformula of A, then so is $B(t)$.

DEFINITION. The notions of *positive, negative, strictly positive* subformula are defined in a similar style:

(a) A is a positive and a strictly positive subformula of itself;
(b) if $B \wedge C$ or $B \vee C$ is a positive (negative, strictly positive) subformula of A, then so are B, C;
(c) if $\forall_x B(x)$ or $\exists_x B(x)$ is a positive (negative, strictly positive) subformula of A, then so is $B(t)$;
(d) if $B \rightarrow C$ is a positive (negative) subformula of A, then B is a negative (positive) subformula of A, and C is a positive (negative) subformula of A;
(e) if $B \rightarrow C$ is a strictly positive subformula of A, then so is C.

A strictly positive subformula of A is also called a *strictly positive part (s.p.p.)* of A. Note that the set of subformulas of A is the union of the positive and negative subformulas of A.

EXAMPLE. $(P \to Q) \to R \wedge \forall_x S(x)$ has as s.p.p.'s the whole formula, $R \wedge \forall_x S(x)$, R, $\forall_x S(x)$, $S(t)$. The positive subformulas are the s.p.p.'s and in addition P; the negative subformulas are $P \to Q$, Q.

1.1.4. Examples of derivations. To motivate the rules for natural deduction, let us start with informal proofs of some simple logical facts.

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

Informal proof. Assume $A \to B \to C$. To show: $(A \to B) \to A \to C$. So assume $A \to B$. To show: $A \to C$. So finally assume A. To show: C. Using the third assumption twice we have $B \to C$ by the first assumption, and B by the second assumption. From $B \to C$ and B we then obtain C. Then $A \to C$, cancelling the assumption on A; $(A \to B) \to A \to C$ cancelling the second assumption; and the result follows by cancelling the first assumption. □

$$\forall_x(A \to B) \to A \to \forall_x B, \quad \text{if } x \notin \mathrm{FV}(A).$$

Informal proof. 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 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. Hence $\forall_x B$. Hence $A \to \forall_x B$, cancelling the second assumption. Hence the result, cancelling the first assumption. □

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 as the proof progresses, free assumptions may become cancelled or "closed" because of the implies-introduction rule.

We 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 each node of which denotes a rule application. The labels of the inner nodes are the formulas derived as conclusions at those points, and the labels of the leaves are formulas or terms. The labels of the nodes immediately above a node k are the *premises* of the rule application. At the root of the tree we have the conclusion (or end formula) of the whole derivation. In natural deduction systems one works with *assumptions* at leaves of the tree; they can be either *open* or *closed* (cancelled). Any of these assumptions carries a *marker*. As markers we use *assumption variables* denoted $u, v, w, u_0, u_1, \ldots$. The variables of the language previously introduced will

now often be called *object variables*, to distinguish them from assumption variables. If at a node below an assumption the dependency on this assumption is removed (it becomes closed) we record this by writing down the assumption variable. Since the same assumption may be used more than once (this was the case in the first example above), the assumption marked with u (written $u\colon A$) may appear many times. Of course we insist that distinct assumption formulas must have distinct markers. An inner node of the tree is understood as the result of passing from premises to the conclusion of 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 or cancels an assumption variable u (and hence removes the dependency of all assumptions $u\colon A$ thus marked). 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.

Definition. A formula A is called *derivable* (in *minimal logic*), written $\vdash A$, if there is a derivation of A (without free assumptions) using the natural deduction rules. 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 $A_1, \dots, A_n \in \Gamma$.

We now formulate the rules of natural deduction.

1.1.5. Introduction and elimination rules for $\to$ and $\forall$. First we have an assumption rule, allowing to write down an arbitrary formula A together with a marker u:

$$u\colon A \qquad \text{assumption.}$$

The other rules of natural deduction split into introduction rules (I-rules for short) and elimination rules (E-rules) for the logical connectives which, for the time being, are just $\to$ and $\forall$. For implication $\to$ there is an introduction rule $\to^+$ and an elimination rule $\to^-$ also called *modus ponens*. The left premise $A \to B$ in $\to^-$ is called the *major* (or *main*) premise, and the right premise A the *minor* (or *side*) premise. Note that with an application of the $\to^+$-rule *all* assumptions above it marked with $u\colon A$ are cancelled (which is denoted by putting square brackets around these assumptions), and the u then gets written alongside. There may of course be other uncancelled assumptions $v\colon A$ of the same formula A, which may get cancelled at a later

stage.

$$\frac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B}\,{\to}^{+}u \qquad \frac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B}\,{\to}^{-}$$

For the universal quantifier $\forall$ there is an introduction rule $\forall^+$ (again marked, but now with the bound variable x) and an elimination rule $\forall^-$ whose right premise is the term t to be substituted. The rule $\forall^+ x$ with conclusion $\forall_x A$ is subject to the following *(eigen-)variable condition*: the derivation M of the premise A must not contain any open assumption having x as a free variable.

$$\frac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A}\,\forall^{+}x \qquad \frac{\begin{array}{cc}\mid M & \\ \forall_x A(x) & t\end{array}}{A(t)}\,\forall^{-}$$

We now give derivations of the two example formulas treated informally above. Since in many cases the rule used is determined by the conclusion, we suppress in such cases the name of the rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon A \to B \to C \qquad w\colon A}{B \to C} \qquad \dfrac{v\colon A \to B \qquad w\colon A}{B}}{C}}{A \to C}\,{\to}^{+}w}{(A \to B) \to A \to C}\,{\to}^{+}v}{(A \to B \to C) \to (A \to B) \to A \to C}\,{\to}^{+}u$$

$$\dfrac{\dfrac{\dfrac{\dfrac{u\colon \forall_x(A \to B) \qquad x}{A \to B} \qquad v\colon A}{B}}{\forall_x B}\,\forall^{+}x}{A \to \forall_x B}\,{\to}^{+}v}{\forall_x(A \to B) \to A \to \forall_x B}\,{\to}^{+}u$$

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

1.1.6. Properties of negation. Recall that negation is defined by $\neg A := (A \to \bot)$. The following can easily be derived.

$$A \to \neg\neg A,$$
$$\neg\neg\neg A \to \neg A.$$

However, $\neg\neg A \to A$ is in general *not* derivable (without stability – we will come back to this later on).

LEMMA. *The following are derivable.*

$$(A \to B) \to \neg B \to \neg A,$$
$$\neg(A \to B) \to \neg B,$$
$$\neg\neg(A \to B) \to \neg\neg A \to \neg\neg B,$$
$$(\bot \to B) \to (\neg\neg A \to \neg\neg B) \to \neg\neg(A \to B),$$
$$\neg\neg\forall_x A \to \forall_x \neg\neg A.$$

Derivations are left as an exercise.

1.1.7. Introduction and elimination rules for disjunction $\vee$, conjunction $\wedge$ and existence $\exists$. For disjunction the introduction and elimination rules are

$$\frac{\begin{array}{c}|\,M\\ A\end{array}}{A \vee B}\vee_0^+ \qquad \frac{\begin{array}{c}|\,M\\ B\end{array}}{A \vee B}\vee_1^+ \qquad \frac{\begin{array}{ccc} & [u\colon A] & [v\colon B]\\ |\,M & |\,N & |\,K\\ A \vee B & C & C\end{array}}{C}\vee^- u, v$$

For conjunction we have

$$\frac{\begin{array}{cc}|\,M & |\,N\\ A & B\end{array}}{A \wedge B}\wedge^+ \qquad \frac{\begin{array}{ccc} & [u\colon A] & [v\colon B]\\ |\,M & & |\,N\\ A \wedge B & & C\end{array}}{C}\wedge^- u, v$$

and for the existential quantifier

$$\frac{\begin{array}{cc} & |\,M\\ t & A(t)\end{array}}{\exists_x A(x)}\exists^+ \qquad \frac{\begin{array}{ccc} & [u\colon A] & \\ |\,M & & |\,N\\ \exists_x A & & B\end{array}}{B}\exists^- x, u \text{ (var.cond.)}$$

Similar to $\forall^+ x$ the rule $\exists^- x, u$ is subject to an *(eigen-)variable condition*: in the derivation N the variable x (i) should not occur free in the formula of any open assumption other than $u\colon A$, and (ii) should not occur free in B.

Again, in each of the elimination rules $\vee^-$, $\wedge^-$ and $\exists^-$ the left premise is called *major* (or *main*) premise, and the right premise is called the *minor* (or *side*) premise.

It is easy to see that for each of the connectives $\vee$, $\wedge$, $\exists$ the rules and the following axioms are equivalent over minimal logic; this is left as an exercise.

For disjunction the introduction and elimination axioms are

$$\begin{aligned}
&\vee_0^+ \colon A \to A \vee B,\\
&\vee_1^+ \colon B \to A \vee B,\\
&\vee^- \colon A \vee B \to (A \to C) \to (B \to C) \to C.
\end{aligned}$$

For conjunction we have

$$\wedge^+ \colon A \to B \to A \wedge B, \qquad \wedge^- \colon A \wedge B \to (A \to B \to C) \to C$$

and for the existential quantifier

$$\exists^+ \colon A \to \exists_x A, \qquad \exists^- \colon \exists_x A \to \forall_x (A \to B) \to B \quad (x \notin \mathrm{FV}(B)).$$

Remark. All these axioms can be seen as special cases of a general schema, that of an *inductively defined predicate*, which is defined by some introduction rules and one elimination rule.

We collect some easy facts about derivability; $B \leftarrow A$ means $A \to B$.

Lemma. *The following are derivable.*

$$\begin{aligned}
&(A \wedge B \to C) \leftrightarrow (A \to B \to C),\\
&(A \to B \wedge C) \leftrightarrow (A \to B) \wedge (A \to C),\\
&\\
&(A \vee B \to C) \leftrightarrow (A \to C) \wedge (B \to C),\\
&(A \to B \vee C) \leftarrow (A \to B) \vee (A \to C),\\
&\\
&(\forall_x A \to B) \leftarrow \exists_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),\\
&(A \to \forall_x B) \leftrightarrow \forall_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(A),\\
&\\
&(\exists_x A \to B) \leftrightarrow \forall_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(B),\\
&(A \to \exists_x B) \leftarrow \exists_x (A \to B) \quad \textit{if } x \notin \mathrm{FV}(A).
\end{aligned}$$

Proof. A derivation of the final formula is

$$\dfrac{\dfrac{u \colon \exists_x (A \to B) \qquad \dfrac{\dfrac{\dfrac{}{x} \qquad \dfrac{w \colon A \to B \qquad v \colon A}{B}}{\exists_x B}\, \exists^- x, w}{\exists_x B}}{A \to \exists_x B}\, \to^+ v}{\exists_x (A \to B) \to A \to \exists_x B}\, \to^+ u$$

The variable condition for $\exists^-$ is satisfied since the variable x (i) is not free in the formula A of the open assumption $v \colon A$, and (ii) is not free in $\exists_x B$. The rest of the proof is left as an exercise. □

As already mentioned, we distinguish between two kinds of "exists" and two kinds of "or": the "weak" or classical ones and the "strong" or non-classical ones, with constructive content. In the present context both kinds occur together and hence we must mark the distinction; we shall do this by writing a tilde above the weak disjunction and existence symbols thus

$$A \mathbin{\tilde{\vee}} B := \neg A \to \neg B \to \bot, \qquad \tilde{\exists}_x A := \neg \forall_x \neg A.$$

These weak variants of disjunction and the existential quantifier are no stronger than the proper ones (in fact, they are weaker):

$$A \vee B \to A \mathbin{\tilde{\vee}} B, \qquad \exists_x A \to \tilde{\exists}_x A.$$

This can be seen easily by putting $C := \bot$ in $\vee^-$ and $B := \bot$ in $\exists^-$.

Remark. Since $\tilde{\exists}_x \tilde{\exists}_y A$ unfolds into a rather awkward formula we extend the $\tilde{\exists}$-terminology to lists of variables:

$$\tilde{\exists}_{x_1,\dots,x_n} A := \forall_{x_1,\dots,x_n}(A \to \bot) \to \bot.$$

Moreover let

$$\tilde{\exists}_{x_1,\dots,x_n}(A_1 \mathbin{\tilde{\wedge}} \dots \mathbin{\tilde{\wedge}} A_m) := \forall_{x_1,\dots,x_n}(A_1 \to \dots \to A_m \to \bot) \to \bot.$$

This allows to stay in the $\to, \forall$ part of the language. Notice that $\tilde{\wedge}$ only makes sense in this context, i.e., in connection with $\tilde{\exists}$.

1.1.8. Intuitionistic and classical derivability. In the definition of derivability in Section 1.1.4 falsity $\bot$ plays no role. We may change this and require *ex-falso-quodlibet* axioms, of the form

$$\forall_{\vec{x}}(\bot \to R\vec{x}\,)$$

with R a relation symbol distinct from $\bot$. Let Efq denote the set of all such axioms. A formula A is called *intuitionistically derivable*, written $\vdash_i A$, if $\mathrm{Efq} \vdash A$. We write $\Gamma \vdash_i B$ for $\Gamma \cup \mathrm{Efq} \vdash B$.

We may even go further and require *stability* axioms, of the form

$$\forall_{\vec{x}}(\neg\neg R\vec{x} \to R\vec{x}\,)$$

with R again a relation symbol distinct from $\bot$. Let Stab denote the set of all these axioms. A formula A is called *classically derivable*, written $\vdash_c A$, if $\mathrm{Stab} \vdash A$. We write $\Gamma \vdash_c B$ for $\Gamma \cup \mathrm{Stab} \vdash B$.

It is easy to see that intuitionistically (i.e., from Efq) we can derive $\bot \to A$ for an *arbitrary* formula A, using the introduction rules for the connectives. A similar generalization of the stability axioms is only possible for formulas in the language not involving $\vee, \exists$. However, it is still possible to use the substitutes $\tilde{\vee}$ and $\tilde{\exists}$.

Theorem (Stability, or principle of indirect proof).

(a) $\vdash (\neg\neg A \to A) \to (\neg\neg B \to B) \to \neg\neg(A \wedge B) \to A \wedge B$.
(b) $\vdash (\neg\neg B \to B) \to \neg\neg(A \to B) \to A \to B$.
(c) $\vdash (\neg\neg A \to A) \to \neg\neg\forall_x A \to A$.
(d) $\vdash_c \neg\neg A \to A$ *for every formula* A *without* $\vee, \exists$.

PROOF. (a) is left as an exercise.

(b) For simplicity, in the derivation to be constructed we leave out applications of $\to^+$ at the end.

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

(c)

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

(d) Induction on A. The case $R\vec{t}$ with R distinct from $\bot$ is given by Stab. In the case $\bot$ the desired derivation is

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

In the cases $A \wedge B$, $A \to B$ and $\forall_x A$ use (a), (b) and (c), respectively. □

Using stability we can prove some well-known facts about the interaction of weak disjunction and the weak existential quantifier with implication. We first prove a more refined claim, stating to what extent we need to go beyond minimal logic.

LEMMA. *The following are derivable.*

(1) $(\tilde{\exists}_x A \to B) \to \forall_x(A \to B)$ *if* $x \notin \mathrm{FV}(B)$,

(2) $(\neg\neg B \to B) \to \forall_x(A \to B) \to \tilde{\exists}_x A \to B$ *if* $x \notin \mathrm{FV}(B)$,

(3) $(\bot \to B[x{:=}c]) \to (A \to \tilde{\exists}_x B) \to \tilde{\exists}_x(A \to B)$ *if* $x \notin \mathrm{FV}(A)$,

(4) $\tilde{\exists}_x(A \to B) \to A \to \tilde{\exists}_x B$ *if* $x \notin \mathrm{FV}(A)$.

The last two items can also be seen as simplifying a weakly existentially quantified implication whose premise does not contain the quantified variable. In case the conclusion does not contain the quantified variable we have

(5) $(\neg\neg B \to B) \to \quad \tilde{\exists}_x(A \to B) \to \forall_x A \to B \qquad$ *if* $x \notin \mathrm{FV}(B)$,

(6) $\forall_x(\neg\neg A \to A) \to (\forall_x A \to B) \to \tilde{\exists}_x(A \to B) \quad$ *if* $x \notin \mathrm{FV}(B)$.

Proof. (1)

$$\dfrac{\tilde{\exists}_x A \to B \qquad \dfrac{\dfrac{u_1 \colon \forall_x \neg A \qquad x}{\neg A} \qquad A}{\bot}}{\neg\forall_x \neg A} \to^+ u_1}{B}$$

(2)

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\dfrac{\neg\forall_x \neg A \qquad \dfrac{\dfrac{\dfrac{\forall_x(A \to B) \qquad x}{A \to B} \qquad u_1 \colon A}{B}}{\bot}}{\neg A}}{\forall_x \neg A}}{\bot} \to^+ u_1}{\neg\neg B} \to^+ u_2}{B}$$

(3) Writing B_0 for $B[x{:=}c]$ we have

$$\dfrac{\dfrac{\forall_x \neg(A \to B) \quad c}{\neg(A \to B_0)} \qquad \dfrac{\dfrac{\bot \to B_0 \qquad \dfrac{\dfrac{A \to \tilde{\exists}_x B \quad u_2 \colon A}{\tilde{\exists}_x B} \qquad \dfrac{\dfrac{\dfrac{\forall_x \neg(A \to B) \quad x}{\neg(A \to B)} \qquad \dfrac{u_1 \colon B}{A \to B}}{\bot}}{\neg B}}{\forall_x \neg B}}{\bot}}{B_0}}{A \to B_0} \to^+ u_2}{\bot}$$

(4)

$$\dfrac{\tilde{\exists}_x(A \to B) \qquad \dfrac{\dfrac{\dfrac{\forall_x \neg B \qquad x}{\neg B} \qquad \dfrac{u_1 \colon A \to B \qquad A}{B}}{\bot}}{\neg(A \to B)}}{\forall_x \neg(A \to B)} \to^+ u_1}{\bot}$$

(5)

$$\dfrac{\neg\neg B \to B \qquad \dfrac{\dfrac{\tilde\exists_x(A \to B) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\forall_x A \quad x}{A} \qquad \overset{u_1}{\dfrac{A \to B \qquad B}{\bot}}}{\neg(A \to B)}\to^+ u_1}{\forall_x \neg(A \to B)}}{\bot}}{\neg\neg B}\to^+ u_2}{B}$$

(6) We derive $\forall_x(\bot \to A) \to (\forall_x A \to B) \to \forall_x \neg(A \to B) \to \neg\neg A$. Writing Ax, Ay for $A(x), A(y)$ we have

$$\dfrac{\dfrac{\forall_x \neg(Ax \to B) \quad x}{\neg(Ax \to B)} \qquad \dfrac{\forall_x Ax \to B \qquad \dfrac{\dfrac{\dfrac{\forall_y(\bot \to Ay) \quad y}{\bot \to Ay} \qquad \overset{u_1}{\dfrac{\neg Ax \quad u_2}{\bot}}}{Ay}}{\forall_y Ay}}{\dfrac{B}{Ax \to B}}\to^+ u_2}{\bot}}{\neg\neg Ax}\to^+ u_1$$

Using this derivation M we obtain

$$\dfrac{\dfrac{\forall_x \neg(Ax \to B) \quad x}{\neg(Ax \to B)} \qquad \dfrac{\forall_x Ax \to B \qquad \dfrac{\dfrac{\dfrac{\forall_x(\neg\neg Ax \to Ax) \quad x}{\neg\neg Ax \to Ax} \qquad \overset{M}{\neg\neg Ax}}{Ax}}{\forall_x Ax}}{\dfrac{B}{Ax \to B}}}{\bot}$$

Since clearly $\vdash (\neg\neg A \to A) \to \bot \to A$ the claim follows. □

Remark. An immediate consequence of (6) is the classical derivability of the "drinker formula" $\tilde\exists_x(Px \to \forall_x Px)$, to be read "in every non-empty bar there is a person such that, if this person drinks, then everybody drinks". To see this let $A := Px$ and $B := \forall_x Px$ in (6).

Corollary.

$\vdash_c (\tilde\exists_x A \to B) \leftrightarrow \forall_x(A \to B)$ *if* $x \notin \mathrm{FV}(B)$ *and* B *without* $\vee, \exists$,

$\vdash_i (A \to \tilde\exists_x B) \leftrightarrow \tilde\exists_x(A \to B)$ *if* $x \notin \mathrm{FV}(A)$,

$\vdash_c \tilde\exists_x(A \to B) \leftrightarrow (\forall_x A \to B)$ *if* $x \notin \mathrm{FV}(B)$ *and* A, B *without* $\vee, \exists$.

There is a similar lemma on weak disjunction:

LEMMA. *The following are derivable.*

(7) $$(A \mathbin{\tilde\vee} B \to C) \to (A \to C) \wedge (B \to C),$$

(8) $$(\neg\neg C \to C) \to (A \to C) \to (B \to C) \to A \mathbin{\tilde\vee} B \to C,$$

(9) $$(\bot \to B) \to \quad (A \to B \mathbin{\tilde\vee} C) \to (A \to B) \mathbin{\tilde\vee} (A \to C),$$

(10) $$(A \to B) \mathbin{\tilde\vee} (A \to C) \to A \to B \mathbin{\tilde\vee} C,$$

(11) $$(\neg\neg C \to C) \to (A \to C) \mathbin{\tilde\vee} (B \to C) \to A \to B \to C,$$

(12) $$(\bot \to C) \to \quad (A \to B \to C) \to (A \to C) \mathbin{\tilde\vee} (B \to C).$$

PROOF. We only consider (8) and (12); the rest is left as an exercise.
(8)

$$\dfrac{\neg\neg C \to C \qquad \dfrac{\dfrac{\neg A \to \neg B \to \bot \qquad \dfrac{\dfrac{A \to C \quad u_2 : A}{C}}{\bot}\,{\to^+}u_2}{\neg A}}{\neg B \to \bot} \qquad \dfrac{\dfrac{\dfrac{B \to C \quad u_3 : B}{C}}{\bot}\,{\to^+}u_3}{\neg B}}{\bot}\,{\to^+}u_1}{\neg\neg C}}{C}$$

(12)

$$\dfrac{\neg(B \to C) \qquad \dfrac{\dfrac{\bot \to C \qquad \dfrac{\dfrac{\neg(A \to C) \qquad \dfrac{\dfrac{A \to B \to C \quad u_1 : A}{B \to C} \qquad u_2 : B}{C}}{A \to C}\,{\to^+}u_1}{\bot}}{C}}{B \to C}\,{\to^+}u_2}{\bot}$$

The general idea here is to view $\tilde\vee$ as a finitary version of $\tilde\exists$. □

COROLLARY.

$\vdash_c (A \mathbin{\tilde\vee} B \to C) \leftrightarrow (A \to C) \wedge (B \to C)$ *for* C *without* $\vee, \exists$,
$\vdash_i (A \to B \mathbin{\tilde\vee} C) \leftrightarrow (A \to B) \mathbin{\tilde\vee} (A \to C)$,
$\vdash_c (A \to C) \mathbin{\tilde\vee} (B \to C) \leftrightarrow (A \to B \to C)$ *for* C *without* $\vee, \exists$.

The weak existential quantifier $\tilde\exists$ and weak disjunction $\tilde\vee$ satisfy the same introduction axioms as the strong ones: this follows from the derivability of $\exists_x A \to \tilde\exists_x A$ and $A \vee B \to A \mathbin{\tilde\vee} B$ (see Section 1.1.7). They also satisfy

the same elimination axioms, provided one restricts the conclusion to stable formulas. For $\tilde{\exists}$ this has been proved in (2), and for $\tilde{\vee}$ in (8).

Therefore when proving a stable goal in minimal logic more proof techniques are available than in the general case. For instance, case distinction on an arbitrary formula A is possible by (8), since $A \,\tilde{\vee}\, \neg A$ is (easily) derivable. Another important example is

LEMMA. *The following is derivable.*

$$\forall_x \neg A \,\tilde{\vee}\, \tilde{\exists}_x A.$$

PROOF. Unfolding $\tilde{\vee}$ and $\tilde{\exists}$ gives

$$(\forall_x(A \to \bot) \to \bot) \to (\underbrace{(\forall_x(A \to \bot) \to \bot)}_{\tilde{\exists}_x A} \to \bot) \to \bot. \qquad \square$$

It is often helpful to use this lemma in a slightly more general form, for instancce

$$\forall_{x,y}(A \to B \to \bot) \,\tilde{\vee}\, \tilde{\exists}_{x,y}(A \,\tilde{\wedge}\, B).$$

The proof is again immediate, since the right hand side $\tilde{\exists}_{x,y}(A \,\tilde{\wedge}\, B)$ unfolds into the negated left hand side.

1.1.9. Gentzen translation. Classical derivability $\Gamma \vdash_c B$ was defined in Section 1.1.8 by $\Gamma \cup \mathrm{Stab} \vdash B$. This embedding of classical logic into minimal logic can be expressed in a somewhat different and very explicit form, namely as a syntactic translation $A \mapsto A^g$ of formulas such that A is derivable in classical logic if and only if its translation A^g is derivable in minimal logic.

DEFINITION (Gentzen translation A^g).

$$\begin{aligned}
(R\vec{t})^g &:= \neg\neg R\vec{t} \quad \text{for } R \text{ distinct from } \bot,\\
\bot^g &:= \bot,\\
(A \vee B)^g &:= A^g \,\tilde{\vee}\, B^g,\\
(\exists_x A)^g &:= \tilde{\exists}_x A^g,\\
(A \circ B)^g &:= A^g \circ B^g \quad \text{for } \circ = \to, \wedge,\\
(\forall_x A)^g &:= \forall_x A^g.
\end{aligned}$$

LEMMA (Stability of A^g). $\vdash \neg\neg A^g \to A^g$.

PROOF. Induction on A.

Case $R\vec{t}$ with R distinct from $\bot$. We must show $\neg\neg\neg\neg R\vec{t} \to \neg\neg R\vec{t}$, which is a special case of $\vdash \neg\neg\neg B \to \neg B$.

Case $\bot$. Use $\vdash \neg\neg\bot \to \bot$.

Case $A \vee B$. We must show $\vdash \neg\neg(A^g \mathbin{\tilde\vee} B^g) \to A^g \mathbin{\tilde\vee} B^g$, which is a special case of $\vdash \neg\neg(\neg C \to \neg D \to \bot) \to \neg C \to \neg D \to \bot$:

$$\dfrac{\neg\neg(\neg C \to \neg D \to \bot) \qquad \dfrac{\dfrac{\dfrac{u_1 \colon \neg C \to \neg D \to \bot \qquad \neg C}{\neg D \to \bot} \qquad \neg D}{\bot}}{\neg(\neg C \to \neg D \to \bot)} \to^+ u_1}{\bot}$$

Case $\exists_x A$. In this case we must show $\vdash \neg\neg\tilde\exists_x A^g \to \tilde\exists_x A^g$, but this is a special case of $\vdash \neg\neg\neg B \to \neg B$, because $\tilde\exists_x A^g$ is the negation $\neg\forall_x \neg A^g$.

Case $A \wedge B$. We must show $\vdash \neg\neg(A^g \wedge B^g) \to A^g \wedge B^g$. By induction hypothesis $\vdash \neg\neg A^g \to A^g$ and $\vdash \neg\neg B^g \to B^g$. Now use part (a) of the stability theorem in Section 1.1.8.

The cases $A \to B$ and $\forall_x A$ are similar, using parts (b) and (c) of the stability theorem instead. □

THEOREM. (a) $\Gamma \vdash_c A$ *implies* $\Gamma^g \vdash A^g$.
(b) $\Gamma^g \vdash A^g$ *implies* $\Gamma \vdash_c A$ *for* Γ, A *without* $\vee, \exists$.

PROOF. (a) We use induction on $\Gamma \vdash_c A$. In case of a stability axiom $\forall_{\vec x}(\neg\neg R\vec x \to R\vec x)$ we must derive $\forall_{\vec x}(\neg\neg\neg\neg R\vec x \to \neg\neg R\vec x)$, which is easy (as above). For the rules $\to^+$, $\to^-$, $\forall^+$, $\forall^-$, $\wedge^+$ and $\wedge^-$ the claim follows immediately from the induction hypothesis, using the same rule again. This works because the Gentzen translation acts as a homomorphism for these connectives. For the rules $\vee_i^+$, $\vee^-$, $\exists^+$ and $\exists^-$ the claim follows from the induction hypothesis and the remark on the elimination rules for $\tilde\vee, \tilde\exists$ in Section 1.1.8. For example, in case $\exists^-$ the induction hypothesis gives

$$\begin{array}{c} \\ \mid M \\ \tilde\exists_x A^g \end{array} \quad \text{and} \quad \begin{array}{c} u \colon A^g \\ \mid N \\ B^g \end{array}$$

with $x \notin \mathrm{FV}(B^g)$. Now use $\vdash (\neg\neg B^g \to B^g) \to \tilde\exists_x A^g \to \forall_x(A^g \to B^g) \to B^g$. Its premise $\neg\neg B^g \to B^g$ is derivable by the lemma above.

(b) First note that $\vdash_c (B \leftrightarrow B^g)$ if B is without $\vee, \exists$. Now assume that Γ, A are without $\vee, \exists$. From $\Gamma^g \vdash A^g$ we obtain $\Gamma \vdash_c A$ as follows. We argue informally. Assume Γ. Then Γ^g by the note, hence A^g because of $\Gamma^g \vdash A^g$, hence A again by the note. □

1.2. Normal derivations

A derivation in normal form does not make "detours", or more precisely, it cannot occur that an elimination rule immediately follows an introduction rule. We use "conversions" to remove such "local maxima" of complexity, thus reducing any given derivation to normal form. We also analyse the

shape of derivations in normal form, and prove the *subformula property*, which says that every formula in a normal derivation is a subformula of the end-formula or else of an assumption. For simplicity we consider derivations involving $\to, \forall$-rules only.

1.2.1. The Curry-Howard correspondence. Since natural deduction derivations can be notationally cumbersome, it will be convenient to represent them as typed "derivation terms", where the derived formula is the "type" of the term (and displayed as a superscript). This representation goes under the name of *Curry-Howard correspondence*. It dates back to Curry (1930) and somewhat later Howard, published only in (1980), who noted that the types of the combinators used in combinatory logic are exactly the Hilbert style axioms for minimal propositional logic. Subsequently Martin-Löf (1972) transferred these ideas to a natural deduction setting where natural deduction proofs of formulas A now correspond exactly to lambda terms with type A. This representation of natural deduction proofs will henceforth be used consistently.

We give an inductive definition of such derivation terms for the $\to, \forall$-rules in Table 1 where for clarity we have written the corresponding derivations to the left. This can be extended to the rules for $\vee$, $\wedge$ and $\exists$, but we will not do this here.

Every derivation term carries a formula as its type. However, we shall usually leave these formulas implicit and write derivation terms without them. Note that every derivation term can be written uniquely in one of the forms

$$u\vec{M} \mid \lambda_v M \mid (\lambda_v M)N\vec{L},$$

where u is an assumption variable or assumption constant, v is an assumption variable or object variable, and M, N, L are derivation terms or object terms. Here the final form is not normal: $(\lambda_v M)N\vec{L}$ is called a *β-redex* (for "reducible expression"). It can be reduced by a "conversion". A *conversion* removes a detour in a derivation, i.e., an elimination immediately following an introduction. We consider the following conversions, for derivations written in tree notation and also as derivation terms.

$\to$-conversion.

$$\dfrac{\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B}\to^+ u \qquad \begin{array}{c}\mid N\\ A\end{array}}{B}\to^- \quad \mapsto_\beta \quad \begin{array}{c}\mid N\\ A\\ \mid M\\ B\end{array}$$

or written as derivation terms

$$(\lambda_u M(u^A)^B)^{A\to B}N^A \mapsto_\beta M(N^A)^B.$$

Derivation	Term
$u\colon A$	u^A
$\dfrac{\begin{array}{c}[u\colon A]\\ \mid M\\ B\end{array}}{A \to B} \to^+ u$	$(\lambda_{u^A} M^B)^{A\to B}$
$\dfrac{\begin{array}{cc}\mid M & \mid N\\ A \to B & A\end{array}}{B} \to^-$	$(M^{A\to B} N^A)^B$
$\dfrac{\begin{array}{c}\mid M\\ A\end{array}}{\forall_x A} \forall^+ x$ (with var.cond.)	$(\lambda_x M^A)^{\forall_x A}$ (with var.cond.)
$\dfrac{\begin{array}{cc}\mid M & \\ \forall_x A(x) & t\end{array}}{A(t)} \forall^-$	$(M^{\forall_x A(x)} t)^{A(t)}$

TABLE 1. Derivation terms for $\to$ and $\forall$

The reader familiar with λ-calculus should note that this is nothing other than β-conversion.

$\forall$-*conversion.*

$$\dfrac{\dfrac{\begin{array}{c}\mid M\\ A(x)\end{array}}{\forall_x A(x)} \forall^+ x \qquad t}{A(t)} \forall^- \quad \mapsto_\beta \quad \begin{array}{c}\mid M'\\ A(t)\end{array}$$

or written as derivation terms

$$(\lambda_x M(x)^{A(x)})^{\forall_x A(x)} t \mapsto_\beta M(t).$$

The *closure* $\mapsto$ of the conversion relation $\mapsto_\beta$ is defined by

(a) If $M \mapsto_\beta M'$, then $M \mapsto M'$.
(b) If $M \mapsto M'$, then also $MN \mapsto M'N$, $NM \mapsto NM'$, $\lambda_v M \mapsto \lambda_v M'$ (*inner reductions*).

Therefore $M \mapsto N$ means that M *reduces in one step to* N, i.e., N is obtained from M by replacement of (an occurrence of) a redex M' of M by a conversum M'' of M', i.e., by a single conversion.

EXAMPLE. Consider assumption variables

$$\begin{array}{lll} x\colon A \to (B \to A) \to A & u\colon A & u'\colon A \\ y\colon A \to B \to A & v\colon B \to A & v'\colon B \\ z\colon A & & \end{array}$$

Then we have derivation terms

$$\begin{aligned} S &:= \lambda_x\lambda_y\lambda_z(xz(yz))^A &&\colon (A \to (B \to A) \to A) \to (A \to B \to A) \to A \to A \\ K &:= \lambda_u\lambda_v u &&\colon A \to (B \to A) \to A \\ K' &:= \lambda_{u'}\lambda_{v'}u' &&\colon A \to B \to A \end{aligned}$$

By the one step reduction relation we obtain

$$\begin{aligned} SKK' \mapsto\ & (\lambda_x\lambda_y\lambda_z(xz(yz)))(\lambda_u\lambda_v\, u)(\lambda_{u'}\lambda_{v'}\, u') \mapsto \\ & (\lambda_y\lambda_z((\lambda_u\lambda_v\, u)z(yz)))(\lambda_{u'}\lambda_{v'}\, u') \quad \mapsto \\ & (\lambda_y\lambda_z((\lambda_v\, z)(yz)))(\lambda_{u'}\lambda_{v'}\, u') \quad \mapsto \\ & (\lambda_y\lambda_z\, z)(\lambda_{u'}\lambda_{v'}\, u') \quad \mapsto \lambda_z\, z. \end{aligned}$$

The relation $\mapsto^+$ ("*properly reduces to*") is the transitive closure of $\mapsto$, and $\mapsto^*$ ("*reduces to*") is the reflexive and transitive closure of $\mapsto$. The relation $\mapsto^*$ is said to be the notion of reduction *generated* by $\mapsto$.

LEMMA (Substitutivity of $\mapsto$).
(a) *If* $M(v) \mapsto M'(v)$, *then* $M(N) \mapsto M'(N)$.
(b) *If* $N \mapsto N'$, *then* $M(N) \mapsto^* M(N')$.

PROOF. (a) is proved by induction on $M(v) \mapsto M'(v)$; (b) by induction on $M(v)$. Notice that the reason for $\mapsto^*$ in (b) is the fact that v may have many occurrences in $M(v)$. □

A term M is *in normal form*, or M is *normal*, if M does not contain a redex. M *has a normal form* if there is a normal N such that $M \mapsto^* N$. A *reduction sequence* is a (finite or infinite) sequence $M_0 \mapsto M_1 \mapsto M_2 \ldots$ such that $M_i \mapsto M_{i+1}$, for all i. Finite reduction sequences are partially ordered under the initial part relation; the collection of finite reduction sequences starting from a term M forms a tree, the *reduction tree* of M. The branches of this tree may be identified with the collection of all infinite and

all terminating finite reduction sequences. A term is *strongly normalizing* if its reduction tree is finite.

One can show that every term is strongly normalizing, and that its normal form is uniquely determined.

1.2.2. The structure of normal derivations. To analyze normal derivations, it will be useful to introduce the notion of a *track* in a proof tree, which makes sense for non-normal derivations as well.

DEFINITION. A *track* of a derivation M is a sequence of formula occurrences (f.o.) $A_0, \dots, A_n$ such that

(a) A_0 is a top f.o. in M;
(b) A_i for $i < n$ is not the minor premise of an instance of $\to^-$, and A_{i+1} is directly below A_i;
(c) A_n is either the minor premise of an instance of $\to^-$, or the conclusion of M.

The *track of order* 0, or *main track*, in a derivation is the (unique) track ending in the conclusion of the whole derivation. A *track of order* $n+1$ is a track ending in the minor premise of an $\to^-$-application, with major premise belonging to a track of order n.

LEMMA. *In a derivation each formula occurrence belongs to some track.*

PROOF. By induction on derivations. □

Now consider a normal derivation M. Since by normality an E-rule cannot have the conclusion of an I-rule as its major premise, the E-rules have to precede the I-rules in a track, so the following is obvious: a track may be divided into an E-part, say $A_0, \dots, A_{i-1}$, a minimal formula A_i, and an I-part $A_{i+1}, \dots, A_n$. In the E-part all rules are E-rules; in the I-part all rules are I-rules; A_i is the conclusion of an E-rule and, if $i < n$, a premise of an I-rule. Tracks are pieces of branches of the tree with successive f.o.'s in the subformula relationship: either A_{i+1} is a subformula of A_i or vice versa. As a result, all formulas in a track $A_0, \dots, A_n$ are subformulas of A_0 or of A_n; and from this, by induction on the order of tracks, we see that every formula in M is a subformula either of an open assumption or of the conclusion. To summarize:

THEOREM (Subformula property). *In a normal derivation each formula is a subformula of either the end formula or else an assumption formula.*

PROOF. One proves this for tracks of order n, by induction on n. □

REMARK (Long normal form). The minimal formula in a track can be an implication $A \to B$ or a generalization $\forall_x A$. However, we can apply an

"η-expansion" and replace the occurrence of $A \to B$ or $\forall_x A$ by

$$\dfrac{\dfrac{A \to B \qquad u\colon A}{B}\,{\to}^{-}}{A \to B}\,{\to}^{+}u \qquad \dfrac{\dfrac{\forall_x A \qquad x}{A}\,\forall^{-}}{\forall_x A}\,\forall^{+}x$$

Repeating this process we obtain a derivation in "long normal form", all of whose minimal formulas are neither implications nor generalizations.

1.3. Soundness and completeness for tree models

It is an obvious question to ask whether the logical rules we have been considering suffice, i.e., whether we have forgotten some necessary rules. To answer this question we first have to fix the *meaning* of a formula, i.e., provide a semantics. This will be done by means of the tree models introduced by Beth (1956). Using this concept of a model we will prove soundness and completeness.

1.3.1. Tree models. Consider a finitely branching tree of "possible worlds". The worlds are represented as nodes in this tree. They may be thought of as possible states such that all nodes "above" a node k are the ways in which k may develop in the future. The worlds are increasing; that is, if an atomic formula $R\vec{t}$ is true in a world k, then $R\vec{t}$ is true in all future worlds k'.

More formally, each tree model is based on a finitely branching tree T. A *node* k over a set S is a finite sequence $k = \langle a_0, a_1, \dots, a_{n-1}\rangle$ of elements of S; $\mathrm{lh}(k)$ is the length of k. We write $k \preceq k'$ if k is an initial segment of k'. A *tree* on S is a set of nodes closed under initial segments. A tree T is *finitely branching* if every node in T has finitely many immediate successors. A tree T is *infinite* if for every $n \in \mathbb{N}$ there is a node $k \in T$ such that $\mathrm{lh}(k) = n$. A *branch* of a tree T is a linearly ordered subtree of T with the same root, and a *leaf* of T is a node without successors in T. A tree T is *complete* if every node in T has an immediate successor, i.e., T has no leaves.

For the proof of the completeness theorem, the full tree over $\{0, 1\}$ (whose branches constitute Cantor space) will suffice. The nodes will be all the finite sequences of 0's and 1's, and the ordering is as above. The root is the empty sequence and $k0$ is the sequence k with the element 0 added at the end; similarly for $k1$.

For the rest of this section, fix a countable formal language $\mathcal{L}$.

Definition. Let T be a finitely branching tree. A *tree model* on T is a triple $\mathcal{T} = (D, I_0, I_1)$ such that

(a) D is a non-empty set;
(b) for every n-ary function symbol f (in the underlying language $\mathcal{L}$), I_0 assigns to f a map $I_0(f)\colon D^n \to D$;

(c) for every n-ary relation symbol R and every node $k \in T$, $I_1(R,k) \subseteq D^n$ is assigned in such a way that monotonicity is preserved:

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

If $n = 0$, then $I_1(R,k)$ is either true or false. There is no special requirement set on $I_1(\bot,k)$. (Recall that minimal logic places no particular constraints on falsum $\bot$.) We write $R^{\mathcal{T}}(\vec{a},k)$ for $\vec{a} \in I_1(R,k)$, and $|\mathcal{T}|$ to denote the domain D.

It is obvious from the definition that any tree T can be extended to a complete tree $\bar{T}$ (i.e., without leaves), in which for every leaf $k \in T$ all sequences $k0$, $k00$, $k000, \dots$ are added to T. For every node $k0\dots0$, we then add $I_1(R,k0\dots0) := I_1(R,k)$. In the sequel we assume that all trees T are complete.

An *assignment* (or variable assignment) in D is a map η assigning to every variable $x \in \mathrm{dom}(\eta)$ a value $\eta(x) \in D$. Finite assignments will be written as $[x_1 := a_1, \dots, x_n := a_n]$ or else as $[a_1/x_1, \dots, a_n/x_n]$, with distinct $x_1, \dots, x_n$. If η is an assignment in D and $a \in D$, let η^a_x be the assignment in D mapping x to a and coinciding with η elsewhere:

$$\eta^a_x(y) := \begin{cases} \eta(y) & \text{if } y \neq x, \\ a & \text{if } y = x. \end{cases}$$

Let a tree model $\mathcal{T} = (D, I_0, I_1)$ and an assignment η in D be given. We define a homomorphic extension of η (denoted by η as well) to terms t whose variables lie in $\mathrm{dom}(\eta)$ by

$$\begin{aligned} \eta(c) &:= I_0(c), \\ \eta(f(t_1,\dots,t_n)) &:= I_0(f)(\eta(t_1),\dots,\eta(t_n)). \end{aligned}$$

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

Definition. $\mathcal{T}, k \Vdash A[\eta]$ ($\mathcal{T}$ *forces* A at node k for an assignment η) is defined inductively. We write $k \Vdash A[\eta]$ when it is clear from the context what the underlying model $\mathcal{T}$ is, and $\forall_{k' \succeq_n k} A$ for $\forall_{k' \succeq k}(\mathrm{lh}(k') = \mathrm{lh}(k) + n \to A)$.

$$\begin{aligned} k \Vdash (R\vec{t})[\eta] &:= \exists_n \forall_{k' \succeq_n k} R^{\mathcal{T}}(\vec{t}^{\,\mathcal{T}}[\eta], k'), \\ k \Vdash (A \vee B)[\eta] &:= \exists_n \forall_{k' \succeq_n k}(k' \Vdash A[\eta] \vee k' \Vdash B[\eta]), \\ k \Vdash (\exists_x A)[\eta] &:= \exists_n \forall_{k' \succeq_n k} \exists_{a \in |\mathcal{T}|}(k' \Vdash A[\eta^a_x]), \\ k \Vdash (A \to B)[\eta] &:= \forall_{k' \succeq k}(k' \Vdash A[\eta] \to k' \Vdash B[\eta]), \\ k \Vdash (A \wedge B)[\eta] &:= k \Vdash A[\eta] \wedge k \Vdash B[\eta], \\ k \Vdash (\forall_x A)[\eta] &:= \forall_{a \in |\mathcal{T}|}(k \Vdash A[\eta^a_x]). \end{aligned}$$

Thus in the atomic, disjunctive and existential cases, the set of k' whose length is $\mathrm{lh}(k)+n$ acts as a "bar" in the complete tree. Note that the implicational case is treated differently, and refers to the "unbounded future".

In this definition, the logical connectives $\to, \wedge, \vee, \forall, \exists$ on the left hand side are part of the object language, whereas the same connectives on the right hand side are to be *understood* in the usual sense: they belong to the "metalanguage". It should always be clear from the context whether a formula is part of the object or the metalanguage.

1.3.2. Covering lemma. It is easily seen (using the definition and monotonicity) that from $k \Vdash A[\eta]$ and $k \preceq k'$ we can conclude $k' \Vdash A[\eta]$. The converse is true as well:

Lemma (Covering).

$$\forall_{k' \succeq_n k}(k' \Vdash A[\eta]) \to k \Vdash A[\eta].$$

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

Case $R\vec{t}$. Assume

$$\forall_{k' \succeq_n k}(k' \Vdash R\vec{t}\,),$$

hence by definition

$$\forall_{k' \succeq_n k}\exists_m \forall_{k'' \succeq_m k'} R^{\mathcal{T}}(\vec{t}^{\,\mathcal{T}}[\eta], k'').$$

Since T is a finitely branching tree,

$$\exists_m \forall_{k' \succeq_m k} R^{\mathcal{T}}(\vec{t}^{\,\mathcal{T}}[\eta], k').$$

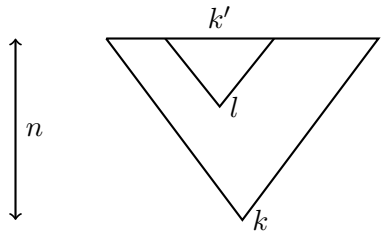
Hence $k \Vdash R\vec{t}$.

The cases $A \vee B$ and $\exists_x A$ are handled similarly.

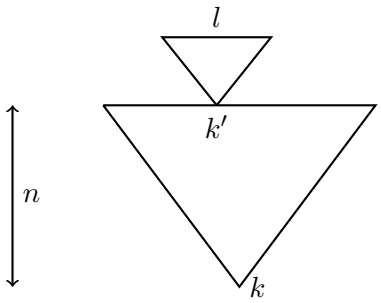
Case $A \to B$. Assume that for all $k' \succeq k$ with $\mathrm{lh}(k') = \mathrm{lh}(k)+n$ we have $k' \Vdash A \to B$. The goal is $k \Vdash A \to B$, i.e.,

$$\forall_{l \succeq k}(l \Vdash A \to l \Vdash B).$$

Let $l \succeq k$ and $l \Vdash A$. We must show $l \Vdash B$. *Case* $\mathrm{lh}(l) \le \mathrm{lh}(k')$:



Then $k' \Vdash B$ for all these k', hence $l \Vdash B$ by IH for B. *Case* $\mathrm{lh}(k') < \mathrm{lh}(l)$:



Because of $k' \Vdash A \to B$ and $l \Vdash A$ we obtain $l \Vdash B$, by definition of $\Vdash$.

The cases $A \wedge B$ and $\forall_x A$ are easy. □

1.3.3. Soundness.

Lemma (Coincidence). *Let $\mathcal{T}$ be a tree model, t a term, A a formula and η, ξ assignments in $|\mathcal{T}|$.*

(a) *If $\eta(x) = \xi(x)$ for all $x \in \mathrm{vars}(t)$, then $\eta(t) = \xi(t)$.*
(b) *If $\eta(x) = \xi(x)$ for all $x \in \mathrm{FV}(A)$, then $\mathcal{T}, k \Vdash A[\eta]$ if and only if $\mathcal{T}, k \Vdash A[\xi]$.*

Proof. Induction on terms and formulas. □

Lemma (Substitution). *Let $\mathcal{T}$ be a tree model, $t, r(x)$ terms, $A(x)$ a formula and η an assignment in $|\mathcal{T}|$. Then*

(a) $\eta(r(t)) = \eta_x^{\eta(t)}(r(x))$.
(b) *$\mathcal{T}, k \Vdash A(t)[\eta]$ if and only if $\mathcal{T}, k \Vdash A(x)[\eta_x^{\eta(t)}]$.*

Proof. Induction on terms and formulas. □

Theorem (Soundness). *Let $\Gamma \cup \{A\}$ be a set of formulas such that $\Gamma \vdash A$. Then, if $\mathcal{T}$ is a tree model, k any node and η an assignment in $|\mathcal{T}|$, it follows that $\mathcal{T}, k \Vdash \Gamma[\eta]$ implies $\mathcal{T}, k \Vdash A[\eta]$.*

Proof. Induction on derivations.

We begin with the axiom schemes $\vee_0^+$, $\vee_1^+$, $\vee^-$, $\wedge^+$, $\wedge^-$, $\exists^+$ and $\exists^-$. $k \Vdash C[\eta]$ is abbreviated $k \Vdash C$, when η is known from the context.

Case $\vee_0^+ \colon A \to A \vee B$. We show $k \Vdash A \to A \vee B$. Assume for $k' \succeq k$ that $k' \Vdash A$. Show: $k' \Vdash A \vee B$. This follows from the definition, since $k' \Vdash A$. The case $\vee_1^+ \colon B \to A \vee B$ is symmetric.

Case $\vee^- \colon A \vee B \to (A \to C) \to (B \to C) \to C$. We show that $k \Vdash A \vee B \to (A \to C) \to (B \to C) \to C$. Assume for $k' \succeq k$ that $k' \Vdash A \vee B$, $k' \Vdash A \to C$ and $k' \Vdash B \to C$ (we can safely assume that k' is the same for all three premises). Show that $k' \Vdash C$. By definition, there

is an n s.t. for all $k'' \succeq_n k'$, $k'' \Vdash A$ or $k'' \Vdash B$. In both cases it follows that $k'' \Vdash C$, since $k' \Vdash A \to C$ and $k' \Vdash B \to C$. By the covering lemma, $k' \Vdash C$.

The cases $\wedge^+$, $\wedge^-$ are easy.

Case $\exists^+$: $A \to \exists_x A$. We show $k \Vdash (A \to \exists_x A)[\eta]$. Assume $k' \succeq k$ and $k' \Vdash A[\eta]$. We show $k' \Vdash (\exists_x A)[\eta]$. Since $\eta = \eta_x^{\eta(x)}$ there is an $a \in |\mathcal{T}|$ (namely $a := \eta(x)$) such that $k' \Vdash A[\eta_x^a]$. Hence, $k' \Vdash (\exists_x A)[\eta]$.

Case $\exists^-$: $\exists_x A \to \forall_x(A \to B) \to B$ and $x \notin \mathrm{FV}(B)$. We show that $k \Vdash (\exists_x A \to \forall_x(A \to B) \to B)[\eta]$. Assume that $k' \succeq k$ and $k' \Vdash (\exists_x A)[\eta]$ and $k' \Vdash \forall_x(A \to B)[\eta]$. We show $k' \Vdash B[\eta]$. By definition, there is an n such that for all $k'' \succeq_n k'$ we have $a \in |\mathcal{T}|$ and $k'' \Vdash A[\eta_x^a]$. From $k' \Vdash \forall_x(A \to B)[\eta]$ it follows that $k'' \Vdash B[\eta_x^a]$, and since $x \notin \mathrm{FV}(B)$, from the coincidence lemma, $k'' \Vdash B[\eta]$. Then, finally, by the covering lemma $k' \Vdash B[\eta]$.

This concludes the treatment of the axioms. We now consider the rules. In case of the assumption rule u: A we have $A \in \Gamma$ and the claim is obvious.

Case $\to^+$. Assume $k \Vdash \Gamma$. We show $k \Vdash A \to B$. Assume $k' \succeq k$ and $k' \Vdash A$. Our goal is $k' \Vdash B$. We have $k' \Vdash \Gamma \cup \{A\}$. Thus, $k' \Vdash B$ by induction hypothesis.

Case $\to^-$. Assume $k \Vdash \Gamma$. The induction hypothesis gives us $k \Vdash A \to B$ and $k \Vdash A$. Hence $k \Vdash B$.

Case $\forall^+$. Assume $k \Vdash \Gamma[\eta]$ and $x \notin \mathrm{FV}(\Gamma)$. We show $k \Vdash (\forall_x A)[\eta]$, i.e., $k \Vdash A[\eta_x^a]$ for an arbitrary $a \in |\mathcal{T}|$. We have

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

Case $\forall^-$. Let $k \Vdash \Gamma[\eta]$. We show that $k \Vdash A(t)[\eta]$. This follows from

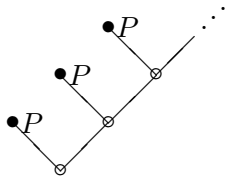
$$\begin{array}{ll} k \Vdash (\forall_x A(x))[\eta] & \text{by induction hypothesis,} \\ k \Vdash A(x)[\eta_x^{\eta(t)}] & \text{by definition,} \\ k \Vdash A(t)[\eta] & \text{by the substitution lemma.} \end{array}$$

This concludes the proof. □

1.3.4. Counter models. With soundness at hand, it is easy to build counter models proving that certain formulas are underivable in minimal or intuitionistic logic. A *tree model for intuitionistic logic* is a tree model $\mathcal{T} = (D, I_0, I_1)$ in which $I_1(\bot, k)$ is false for all k.

As an example we show that $\nvdash_i \neg\neg P \to P$. Assume $\vdash_i \neg\neg P \to P$, i.e., $\mathrm{Efq} \vdash \neg\neg P \to P$. We will obtain a contradiction from this assumption. For simplicity we assume that from Efq we have only used $\bot \to P$ and say $\bot \to Q$ for some Q. We can now substitute $\bot$ for Q everywhere and

obtain a derivation in minimal logic, since $\bot \to \bot$ is immediately derivable. Hence we have $\vdash (\bot \to P) \to \neg\neg P \to P$. We can now obtain the desired contradiction using a tree model determined by the figure below. Next to every node we write all propositions forced at that node.



This is a tree model because monotonicity clearly holds. Observe also that $I_1(\bot, k)$ is false at all nodes k. Hence this is an intuitionistic tree model. By the definition of forcing we have

(i) $\bot \to P$ is forced at every node.
(ii) $P \to \bot$ (i.e., $\neg P$) is never forced.
(iii) $\neg\neg P$ is forced at every node.
(iv) The root node does not force P, since there are arbitrarily long $\circ$-nodes.

This is the desired contradiction to the Soundness Theorem.

The model also shows that the *Peirce formula* $((P \to Q) \to P) \to P$ is not derivable in intuitionistic logic.

1.3.5. Completeness.

THEOREM (Completeness). *Let $\Gamma \cup \{A\}$ be a set of formulas. Then the following propositions are equivalent.*

(a) $\Gamma \vdash A$.
(b) $\Gamma \Vdash A$, *i.e., for all tree models $\mathcal{T}$, nodes k and assignments η*

$$\mathcal{T}, k \Vdash \Gamma[\eta] \to \mathcal{T}, k \Vdash A[\eta].$$

PROOF. Soundness already gives "(a) implies (b)". For the other direction we employ a technique due to Harvey Friedman and construct a tree model $\mathcal{T}$ (over the set T_{01} of all finite 0-1-sequences) whose domain D is the set of all terms of the underlying language, with the property that $\Gamma \vdash B$ is equivalent to $\mathcal{T}, \langle\rangle \Vdash B[\mathrm{id}]$. $\mathcal{T}$ will depend on Γ. We can assume here that Γ and A are closed.

In order to define $\mathcal{T}$, we will need an enumeration $A_0, A_1, A_2, \dots$ of the underlying language $\mathcal{L}$ (assumed countable), in which every formula occurs infinitely often. We also fix an enumeration $x_0, x_1, \dots$ of distinct variables. Since Γ is countable it can we written $\Gamma = \bigcup_n \Gamma_n$ with finite sets Γ_n such that $\Gamma_n \subseteq \Gamma_{n+1}$. With every node $k \in T_{01}$, we associate a finite set Δ_k of formulas and a set V_k of variables, by induction on the length of k.

Let $\Delta_{\langle\rangle} := \emptyset$ and $V_{\langle\rangle} := \emptyset$. Take a node k such that $\mathrm{lh}(k) = n$ and suppose that Δ_k, V_k are already defined. Write $\Delta \vdash_n B$ to mean that there

is a derivation of length $\leq n$ of B from Δ. We define Δ_{k0}, V_{k0} and Δ_{k1}, V_{k1} as follows:

Case 0. $\mathrm{FV}(A_n) \not\subseteq V_k$. Then let

$$\Delta_{k0} := \Delta_{k1} := \Delta_k \quad \text{and} \quad V_{k0} := V_{k1} := V_k.$$

Case 1. $\mathrm{FV}(A_n) \subseteq V_k$ and $\Gamma_n, \Delta_k \nvdash_n A_n$. Let

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

Case 2. $\mathrm{FV}(A_n) \subseteq V_k$ and $\Gamma_n, \Delta_k \vdash_n A_n = A'_n \vee A''_n$. Let

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

Case 3. $\mathrm{FV}(A_n) \subseteq V_k$ and $\Gamma_n, \Delta_k \vdash_n A_n = \exists_x A'_n(x)$. Let

$$\Delta_{k0} := \Delta_{k1} := \Delta_k \cup \{A_n, A'_n(x_i)\} \quad \text{and} \quad V_{k0} := V_{k1} := V_k \cup \{x_i\},$$

where x_i is the first variable $\notin V_k$.

Case 4. $\mathrm{FV}(A_n) \subseteq V_k$ and $\Gamma_n, \Delta_k \vdash_n A_n$, with A_n neither a disjunction nor an existentially quantified formula. Let

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

Obviously $\mathrm{FV}(\Delta_k) \subseteq V_k$, and $k \preceq k'$ implies that $\Delta_k \subseteq \Delta_{k'}$. Notice also that because of $\vdash \exists_x(\bot \to \bot)$ and the fact that this formula is repeated infinitely often in the given enumeration, for every variable x_i there is an m such that $x_i \in V_k$ for all k with $\mathrm{lh}(k) = m$.

We note that

$$(13) \qquad \forall_{k' \succeq_n k} (\Gamma, \Delta_{k'} \vdash B) \to \Gamma, \Delta_k \vdash B, \quad \text{provided } \mathrm{FV}(B) \subseteq V_k.$$

It is sufficient to show that, for $\mathrm{FV}(B) \subseteq V_k$,

$$(\Gamma, \Delta_{k0} \vdash B) \wedge (\Gamma, \Delta_{k1} \vdash B) \to (\Gamma, \Delta_k \vdash B).$$

In cases 0, 1 and 4, this is obvious. For case 2, the claim follows immediately from the axiom schema $\vee^-$. In case 3, we have $\mathrm{FV}(A_n) \subseteq V_k$ and $\Gamma_n, \Delta_k \vdash_n A_n = \exists_x A'_n(x)$. Assume $\Gamma, \Delta_k \cup \{A_n, A'_n(x_i)\} \vdash B$ with $x_i \notin V_k$, and $\mathrm{FV}(B) \subseteq V_k$. Then $x_i \notin \mathrm{FV}(\Delta_k \cup \{A_n, B\})$, hence $\Gamma, \Delta_k \cup \{A_n\} \vdash B$ by $\exists^-$ and therefore $\Gamma, \Delta_k \vdash B$.

Next, we show

$$(14) \qquad \Gamma, \Delta_k \vdash B \to \exists_n \forall_{k' \succeq_n k} (B \in \Delta_{k'}), \quad \text{provided } \mathrm{FV}(B) \subseteq V_k.$$

Choose $n \geq \mathrm{lh}(k)$ such that $B = A_n$ and $\Gamma_n, \Delta_k \vdash_n A_n$. For all $k' \succeq k$, if $\mathrm{lh}(k') = n + 1$ then $A_n \in \Delta_{k'}$ (cf. the cases 2 - 4).

Using the sets Δ_k we can define a tree model $\mathcal{T}$ as (Ter, I_0, I_1) where Ter denotes the set of terms of the underlying language, $I_0(f)(\vec{t}\,) := f\vec{t}$ and

$$R^{\mathcal{T}}(\vec{t}, k) = I_1(R, k)(\vec{t}\,) := (R\vec{t} \in \Delta_k).$$

Obviously, $t^{\mathcal{T}}[\mathrm{id}] = t$ for all terms t.

Now write $k \Vdash B$ for $\mathcal{T}, k \Vdash B[\mathrm{id}]$. We show:

Claim. $\Gamma, \Delta_k \vdash B \leftrightarrow k \Vdash B$ *provided* $\mathrm{FV}(B) \subseteq V_k$.

The proof is by induction on B.

Case $R\vec{t}$. Assume $\mathrm{FV}(R\vec{t}\,) \subseteq V_k$. The following are equivalent:

$$\begin{aligned}
&\Gamma, \Delta_k \vdash R\vec{t}, \\
&\exists_n \forall_{k' \succeq_n k} \, (R\vec{t} \in \Delta_{k'}) && \text{by (14) and (13)}, \\
&\exists_n \forall_{k' \succeq_n k} \, R^{\mathcal{T}}(\vec{t}, k') && \text{by definition of } \mathcal{T}, \\
&k \Vdash R\vec{t} && \text{by definition of } \Vdash, \text{ since } t^{\mathcal{T}}[\mathrm{id}] = t.
\end{aligned}$$

Case $B \vee C$. Assume $\mathrm{FV}(B \vee C) \subseteq V_k$. For the implication $\to$ let $\Gamma, \Delta_k \vdash B \vee C$. Choose an $n \geq \mathrm{lh}(k)$ such that $\Gamma_n, \Delta_k \vdash_n A_n = B \vee C$. Then, for all $k' \succeq k$ s.t. $\mathrm{lh}(k') = n$,

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

and therefore by induction hypothesis

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

Then by definition we have $k \Vdash B \vee C$. For the reverse implication $\leftarrow$ argue as follows.

$$\begin{aligned}
&k \Vdash B \vee C, \\
&\exists_n \forall_{k' \succeq_n k} (k' \Vdash B \vee k' \Vdash C), \\
&\exists_n \forall_{k' \succeq_n k} ((\Gamma, \Delta_{k'} \vdash B) \vee (\Gamma, \Delta_{k'} \vdash C)) && \text{by induction hypothesis}, \\
&\exists_n \forall_{k' \succeq_n k} \, (\Gamma, \Delta_{k'} \vdash B \vee C), \\
&\Gamma, \Delta_k \vdash B \vee C && \text{by (13)}.
\end{aligned}$$

Case $B \wedge C$. This is evident.

Case $B \to C$. Assume $\mathrm{FV}(B \to C) \subseteq V_k$. For $\to$ let $\Gamma, \Delta_k \vdash B \to C$. We must show $k \Vdash B \to C$, i.e.,

$$\forall_{k' \succeq k} (k' \Vdash B \to k' \Vdash C).$$

Let $k' \succeq k$ be such that $k' \Vdash B$. By induction hypothesis, it follows that $\Gamma, \Delta_{k'} \vdash B$. Hence $\Gamma, \Delta_{k'} \vdash C$ follows by assumption. Then again by induction hypothesis $k' \Vdash C$.

For $\leftarrow$ let $k \Vdash B \to C$, i.e., $\forall_{k' \succeq k}(k' \Vdash B \to k' \Vdash C)$. We show that $\Gamma, \Delta_k \vdash B \to C$, using (13). Choose $n \geq \mathrm{lh}(k)$ such that $B = A_n$. For all $k' \succeq_m k$ with $m := n - \mathrm{lh}(k)$ we show that $\Gamma, \Delta_{k'} \vdash B \to C$.

If $\Gamma_n, \Delta_{k'} \vdash_n A_n$, then $k' \Vdash B$ by induction hypothesis, and $k' \Vdash C$ by assumption. Hence $\Gamma, \Delta_{k'} \vdash C$ again by induction hypothesis and thus $\Gamma, \Delta_{k'} \vdash B \to C$.

If $\Gamma_n, \Delta_{k'} \nvdash_n A_n$, then by definition $\Delta_{k'1} = \Delta_{k'} \cup \{B\}$. Hence $\Gamma, \Delta_{k'1} \vdash B$, and thus $k'1 \Vdash B$ by induction hypothesis. Now $k'1 \Vdash C$ by assumption, and finally $\Gamma, \Delta_{k'1} \vdash C$ by induction hypothesis. From $\Delta_{k'1} = \Delta_{k'} \cup \{B\}$ it follows that $\Gamma, \Delta_{k'} \vdash B \to C$.

Case $\forall_x B(x)$. Assume $\mathrm{FV}(\forall_x B(x)) \subseteq V_k$. For $\to$ let $\Gamma, \Delta_k \vdash \forall_x B(x)$. Fix a term t. Then $\Gamma, \Delta_k \vdash B(t)$. Choose $n \geq \mathrm{lh}(k)$ such that $\mathrm{FV}(B(t)) \subseteq V_{k'}$ for all k' with $\mathrm{lh}(k') = n$. Then $\forall_{k' \succeq_m k}(\Gamma, \Delta_{k'} \vdash B(t))$ with $m := n - \mathrm{lh}(k)$, hence $\forall_{k' \succeq_m k}(k' \Vdash B(t))$ by induction hypothesis, hence $k \Vdash B(t)$ by the covering lemma. This holds for every term t, hence $k \Vdash \forall_x B(x)$.

For $\leftarrow$ assume $k \Vdash \forall_x B(x)$. Pick $k' \succeq_n k$ such that $A_m = \exists_x(\bot \to \bot)$, for $m := \mathrm{lh}(k) + n$. Then at height m we put some x_i into the variable sets: for $k' \succeq_n k$ we have $x_i \notin V_{k'}$ but $x_i \in V_{k'j}$. Clearly $k'j \Vdash B(x_i)$, hence $\Gamma, \Delta_{k'j} \vdash B(x_i)$ by induction hypothesis, hence (since at this height we consider the trivial formula $\exists_x(\bot \to \bot)$) also $\Gamma, \Delta_{k'} \vdash B(x_i)$. Since $x_i \notin V_{k'}$ we obtain $\Gamma, \Delta_{k'} \vdash \forall_x B(x)$. This holds for all $k' \succeq_n k$, hence $\Gamma, \Delta_k \vdash \forall_x B(x)$ by (13).

Case $\exists_x B(x)$. Assume $\mathrm{FV}(\exists_x B(x)) \subseteq V_k$. For $\to$ let $\Gamma, \Delta_k \vdash \exists_x B(x)$. Choose an $n \geq \mathrm{lh}(k)$ such that $\Gamma_n, \Delta_k \vdash_n A_n = \exists_x B(x)$. Then, for all $k' \succeq k$ with $\mathrm{lh}(k') = n$

$$\Delta_{k'0} = \Delta_{k'1} = \Delta_{k'} \cup \{\exists_x B(x), B(x_i)\}$$

where $x_i \notin V_{k'}$. Hence by induction hypothesis for $B(x_i)$ (applicable since $\mathrm{FV}(B(x_i)) \subseteq V_{k'j}$ for $j = 0, 1$)

$$k'0 \Vdash B(x_i) \quad \text{and} \quad k'1 \Vdash B(x_i).$$

It follows by definition that $k \Vdash \exists_x B(x)$.

For $\leftarrow$ assume $k \Vdash \exists_x B(x)$. Then $\forall_{k' \succeq_n k} \exists_{t \in \mathrm{Ter}}(k' \Vdash B(x)[\mathrm{id}^t_x])$ for some n, hence $\forall_{k' \succeq_n k} \exists_{t \in \mathrm{Ter}}(k' \Vdash B(t))$. For each of the finitely many $k' \succeq_n k$ pick an m such that $\forall_{k'' \succeq_m k'}(\mathrm{FV}(B(t_{k'})) \subseteq V_{k''})$. Let m_0 be the maximum of all these m. Then

$$\forall_{k'' \succeq_{m_0+n} k} \exists_{t \in \mathrm{Ter}}((k'' \Vdash B(t)) \land \mathrm{FV}(B(t)) \subseteq V_{k''}).$$

The induction hypothesis for $B(t)$ yields

$$\begin{aligned}
&\forall_{k''\succeq_{m_0+n}k}\exists_{t\in\mathrm{Ter}}\,(\Gamma,\Delta_{k''}\vdash B(t)),\\
&\forall_{k''\succeq_{m_0+n}k}\,(\Gamma,\Delta_{k''}\vdash \exists_x B(x)),\\
&\Gamma,\Delta_k\vdash\exists_x B(x) \qquad\qquad \text{by (13)},
\end{aligned}$$

and this completes the proof of the claim.

Now we can finish the proof of the completeness theorem by showing that (b) implies (a). We apply (b) to the tree model $\mathcal{T}$ constructed above from Γ, the empty node $\langle\rangle$ and the assignment $\eta = \mathrm{id}$. Then $\mathcal{T},\langle\rangle \Vdash \Gamma[\mathrm{id}]$ by the claim (since each formula in Γ is derivable from Γ). Hence $\mathcal{T},\langle\rangle \Vdash A[\mathrm{id}]$ by (b) and therefore $\Gamma\vdash A$ by the claim again. □

CHAPTER 2

Recursion Theory

In this chapter we develop the basics of recursive function theory, or as it is more generally known, computability theory. Its history goes back to the seminal works of Turing, Kleene and others in the 1930's.

A computable function is one defined by a program whose operational semantics tell an idealized computer what to do to its storage locations as it proceeds deterministically from input to output, without any prior restrictions on storage space or computation time. We shall be concerned with various program-styles and the relationships between them, but the emphasis throughout will be on one underlying data-type, namely the natural numbers, since it is there that the most basic foundational connections between proof theory and computation are to be seen in their clearest light.

The two best-known models of machine computation are the Turing Machine and the (Unlimited) Register Machine of Shepherdson and Sturgis (1963). We base our development on the latter since it affords the quickest route to the results we want to establish.

2.1. Register machines

2.1.1. Programs. A *register machine* stores natural numbers in registers denoted u, v, w, x, y, z possibly with subscripts, and it responds step by step to a *program* consisting of an ordered list of basic instructions:

$$\begin{array}{l} I_0 \\ I_1 \\ \vdots \\ I_{k-1} \end{array}$$

Each instruction has one of the following three forms whose meanings are obvious:

Zero: $x := 0$,
Succ: $x := x + 1$,
Jump: $[\textbf{if } x = y \textbf{ then } I_n \textbf{ else } I_m]$.

The instructions are obeyed in order starting with I_0 except when a conditional jump instruction is encountered, in which case the next instruction

will be either I_n or I_m according as the numerical contents of registers x and y are equal or not at that stage. The computation *terminates* when it runs out of instructions, that is when the next instruction called for is I_k. Thus if a program of length k contains a jump instruction as above then it must satisfy the condition $n, m \leq k$ and I_k means "halt". Notice of course that some programs do not terminate, for example the following one-liner:

$$[\textbf{if } x = x \textbf{ then } I_0 \textbf{ else } I_1]$$

2.1.2. Program constructs. We develop some shorthand for building up standard sorts of programs.

Transfer. "$x := y$" is the program

$$\begin{array}{l} x := 0 \\ [\textbf{if } x = y \textbf{ then } I_4 \textbf{ else } I_2] \\ x := x + 1 \\ [\textbf{if } x = x \textbf{ then } I_1 \textbf{ else } I_1], \end{array}$$

which copies the contents of register y into register x.

Predecessor. The program "$x := y \dot{-} 1$" copies the modified predecessor of y into x, and simultaneously copies y into z:

$$\begin{array}{l} x := 0 \\ z := 0 \\ [\textbf{if } x = y \textbf{ then } I_8 \textbf{ else } I_3] \\ z := z + 1 \\ [\textbf{if } z = y \textbf{ then } I_8 \textbf{ else } I_5] \\ z := z + 1 \\ x := x + 1 \\ [\textbf{if } z = y \textbf{ then } I_8 \textbf{ else } I_5]. \end{array}$$

Composition. "P **;** Q" is the program obtained by concatenating program P with program Q. However in order to ensure that jump instructions in Q of the form "$[\textbf{if } x = y \textbf{ then } I_n \textbf{ else } I_m]$" still operate properly within Q they need to be re-numbered by changing the addresses n, m to $k+n, k+m$ respectively where k is the length of program P. Thus the effect of this program is to do P until it halts (if ever) and then do Q.

Conditional. "**if** $x = y$ **then** P **else** Q **fi**" is the program

$$\begin{array}{l} [\textbf{if } x = y \textbf{ then } I_1 \textbf{ else } I_{k+2}] \\ \vdots P \\ [\textbf{if } x = x \textbf{ then } I_{k+2+l} \textbf{ else } I_2] \\ \vdots Q \end{array}$$

where k, l are the lengths of the programs P, Q respectively, and again their jump instructions must be appropriately renumbered by adding 1 to the addresses in P and $k+2$ to the addresses in Q. Clearly if $x = y$ then program P is obeyed and the next jump instruction automatically bypasses Q and halts. If $x \neq y$ then program Q is performed.

For Loop. "**for** $i = 1 \dots x$ **do** P **od**" is the program

$$\begin{array}{l} i := 0 \\ [\textbf{if } x = i \textbf{ then } I_{k+4} \textbf{ else } I_2] \\ i := i + 1 \\ \vdots P \\ [\textbf{if } x = i \textbf{ then } I_{k+4} \textbf{ else } I_2] \end{array}$$

where again, k is the length of program P and the jump instructions in P must be appropriately re-addressed by adding 3. The intention of this new program is that it should iterate the program P x times (do nothing if $x = 0$). This requires the restriction that the register x and the "local" counting-register i are not re-assigned new values inside P.

While Loop. "**while** $x \neq 0$ **do** P **od**" is the program

$$\begin{array}{l} y := 0 \\ [\textbf{if } x = y \textbf{ then } I_{k+3} \textbf{ else } I_2] \\ \vdots P \\ [\textbf{if } x = y \textbf{ then } I_{k+3} \textbf{ else } I_2] \end{array}$$

where again, k is the length of program P and the jump instructions in P must be re-addressed by adding 2. This program keeps on doing P until (if ever) the register x becomes 0; it requires the restriction that the auxiliary register y is not re-assigned new values inside P.

2.1.3. Register machine computable functions. A register machine program P may have certain distinguished "input registers" and "output registers". It may also use other "working registers" for scratchwork and these will initially be set to zero. We write $P(x_1, \dots, x_k; y)$ to signify that program P has input registers $x_1, \dots, x_k$ and one output register y, which are distinct.

DEFINITION. The program $P(x_1, \dots, x_k; y)$ is said to *compute* the k-ary partial function $\varphi\colon \mathbb{N}^k \to \mathbb{N}$ if, starting with any numerical values $n_1, \dots, n_k$ in the input registers, the program terminates with the number m in the output register if and only if $\varphi(n_1, \dots, n_k)$ is defined with value m. In this case, the input registers hold their original values.

A function is *register machine computable* if there is some program which computes it.

Here are some examples.

Addition. "Add$(x, y; z)$" is the program

$$z := x \ ; \ \textbf{for } i = 1, \ldots, y \ \textbf{do } z := z + 1 \ \textbf{od}$$

which adds the contents of registers x and y into register z.

Subtraction. "Subt$(x, y; z)$" is the program

$$z := x \ ; \ \textbf{for } i = 1, \ldots, y \ \textbf{do } w := z \dot{-} 1 \ ; \ z := w \ \textbf{od}$$

which computes the modified subtraction function $x \dot{-} y$.

Bounded Sum. If $P(x_1, \ldots, x_k, w; y)$ computes the $k+1$-ary function φ then the program $Q(x_1, \ldots, x_k, z; x)$:

$x := 0$;
for $i = 1, \ldots, z$ **do** $w := i \dot{-} 1$; $P(\vec{x}, w; y)$; $v := x$; Add$(v, y; x)$ **od**

computes the function

$$\psi(x_1, \ldots, x_k, z) = \sum_{w<z} \varphi(x_1, \ldots, x_k, w)$$

which will be undefined if for some $w < z$, $\varphi(x_1, \ldots, x_k, w)$ is undefined.

Multiplication. Deleting "$w := i \dot{-} 1$; P" from the last example gives a program Mult$(z, y; x)$ which places the product of y and z into x.

Bounded Product. If in the bounded sum example, the instruction $x := x + 1$ is inserted immediately after $x := 0$, and if Add$(v, y; x)$ is replaced by Mult$(v, y; x)$, then the resulting program computes the function

$$\psi(x_1, \ldots, x_k, z) = \prod_{w<z} \varphi(x_1, \ldots, x_k, w).$$

Composition. If $P_j(x_1, \ldots, x_k; y_j)$ computes φ_j for each $j = 1, \ldots, n$ and if $P_0(y_1, \ldots, y_n; y_0)$ computes φ_0, then the program $Q(x_1, \ldots, x_k; y_0)$:

$$P_1(x_1, \ldots, x_k; y_1) \ ; \ \ldots \ ; \ P_n(x_1, \ldots, x_k; y_n) \ ; \ P_0(y_1, \ldots, y_n; y_0)$$

computes the function

$$\psi(x_1, \ldots, x_k) = \varphi_0(\varphi_1(x_1, \ldots, x_k), \ldots, \varphi_n(x_1, \ldots, x_k))$$

which will be undefined if any of the φ-subterms on the right hand side is undefined.

Unbounded Minimization. If $P(x_1, \ldots, x_k, y; z)$ computes φ then the program $Q(x_1, \ldots, x_k; z)$:

$y := 0$; $z := 0$; $z := z + 1$;
while $z \neq 0$ **do** $P(x_1, \ldots, x_k, y; z)$; $y := y + 1$ **od** ;
$z := y \dot{-} 1$

computes the function

$$\psi(x_1, \ldots, x_k) \ = \ \mu_y(\varphi(x_1, \ldots, x_k, y) = 0)$$

that is, the *least number* y such that $\varphi(x_1, \ldots, x_k, y')$ is defined for every $y' \leq y$ and $\varphi(x_1, \ldots, x_k, y) = 0$.

2.2. Elementary functions

2.2.1. Definition and simple properties. The *elementary functions* of Kalmár (1943) are those number-theoretic functions which can be defined explicitly by compositional terms built up from variables and the constants $0, 1$ by repeated applications of addition $+$, modified subtraction $\dot{-}$, bounded sums and bounded products.

By omitting bounded products, one obtains the *subelementary* functions.

The examples in the previous section show that all elementary functions are computable and totally defined. Multiplication and exponentiation are elementary since

$$m \cdot n = \textstyle\sum_{i<n} m \text{ and } m^n = \prod_{i<n} m$$

and hence by repeated composition, all exponential polynomials are elementary.

In addition the elementary functions are closed under

Definition by Cases.

$$f(\vec{n}\,) = \begin{cases} g_0(\vec{n}\,) & \text{if } h(\vec{n}\,) = 0 \\ g_1(\vec{n}\,) & \text{otherwise} \end{cases}$$

since f can be defined from g_0, g_1 and h by

$$f(\vec{n}\,) = g_0(\vec{n}\,) \cdot (1 \dot{-} h(\vec{n}\,)) + g_1(\vec{n}\,) \cdot (1 \dot{-} (1 \dot{-} h(\vec{n}\,))).$$

Bounded Minimization.

$$f(\vec{n}, m) = \mu_{k<m}(g(\vec{n}, k) = 0)$$

since f can be defined from g by

$$f(\vec{n}, m) = \sum_{i<m} \big(1 \dot{-} \sum_{k \leq i} (1 \dot{-} g(\vec{n}, k))\big).$$

Note: this definition gives value m if there is no $k < m$ such that $g(\vec{n}, k) = 0$. It shows that not only the elementary, but in fact the subelementary functions are closed under bounded minimization. Furthermore, we define $\mu_{k \leq m}(g(\vec{n}, k) = 0)$ as $\mu_{k<m+1}(g(\vec{n}, k) = 0)$.

Lemma.

(a) *For every elementary function* $f\colon \mathbb{N}^r \to \mathbb{N}$ *there is a number* k *such that for all* $\vec{n} = n_1, \ldots, n_r$,

$$f(\vec{n}\,) < 2_k(\max(\vec{n}\,))$$

where $2_0(m) := m$ *and* $2_{k+1}(m) := 2^{2_k(m)}$.

(b) *The function* $n \mapsto 2_n(1)$ *is not elementary.*

PROOF. (a). By induction on the build-up of the compositional term defining f. The result clearly holds if f is any one of the base functions:

$$f(\vec{n}\,) = 0 \text{ or } 1 \text{ or } n_i \text{ or } n_i + n_j \text{ or } n_i \dot{-} n_j.$$

If f is defined from g by application of bounded sum or product:

$$f(\vec{n}, m) = \sum_{i<m} g(\vec{n}, i) \text{ or } \prod_{i<m} g(\vec{n}, i)$$

where $g(\vec{n}, i) < 2_k(\max(\vec{n}, i))$ then we have

$$f(\vec{n}, m) \leq (2_k(\max(\vec{n}, m)))^m < 2_{k+2}(\max(\vec{n}, m))$$

using $n^n < 2^{2^n}$ (since $n^n < (2^{n-1})^n \leq 2^{2^n}$ for $n \geq 3$).

If f is defined from $g_0, g_1, \ldots, g_l$ by composition:

$$f(\vec{n}\,) = g_0(g_1(\vec{n}\,), \ldots, g_l(\vec{n}\,))$$

where for each $j \leq l$ we have $g_j(-) < 2_{k_j}(\max(-))$, then with $k = \max_j k_j$,

$$f(\vec{n}\,) < 2_k(2_k(\max(\vec{n}\,))) = 2_{2k}(\max(\vec{n}\,))$$

and this completes the first part.

(b). If $2_n(1)$ were an elementary function of n then by (a) there would be a positive k such that for all n,

$$2_n(1) < 2_k(n)$$

but then putting $n = 2_k(1)$ yields $2_{2_k(1)}(1) < 2_{2k}(1)$, a contradiction. □

2.2.2. Elementary relations. A relation R on $\mathbb{N}^k$ is said to be *elementary* if its characteristic function

$$c_R(\vec{n}\,) = \begin{cases} 1 & \text{if } R(\vec{n}\,) \\ 0 & \text{otherwise} \end{cases}$$

is elementary. In particular, the "equality" and "less than" relations are elementary since their characteristic functions can be defined as follows:

$$c_<(n, m) = 1 \dot{-} (1 \dot{-} (m \dot{-} n)), \quad c_=(n, m) = 1 \dot{-} (c_<(n, m) + c_<(m, n)).$$

Furthermore if R is elementary then so is the function

$$f(\vec{n}, m) = \mu_{k<m} R(\vec{n}, k)$$

since $R(\vec{n}, k)$ is equivalent to $1 \dot{-} c_R(\vec{n}, k) = 0$.

LEMMA. *The elementary relations are closed under applications of propositional connectives and bounded quantifiers.*

PROOF. For example, the characteristic function of $\neg R$ is

$$1 \dot{-} c_R(\vec{n}\,).$$

The characteristic function of $R_0 \wedge R_1$ is

$$c_{R_0}(\vec{n}\,) \cdot c_{R_1}(\vec{n}\,).$$

The characteristic function of $\forall_{i<m} R(\vec{n}, i)$ is

$$c_=(m, \mu_{i<m}(c_R(\vec{n}, i) = 0)).$$ □

EXAMPLES. The above closure properties enable us to show that many "natural" functions and relations of number theory are elementary. For instance, it is an easy exercise to show that $\lfloor \frac{n}{m} \rfloor$ is elementary, and then that $n \bmod m$ is elementary. Using this fact we can conclude that the following are elementary as well:

$$\begin{aligned} &\mathrm{Prime}(n) \leftrightarrow 1 < n \wedge \neg\exists_{m<n}(1 < m \wedge n \bmod m = 0), \\ &p_n \qquad\qquad = \mu_{m<2^{2^n}}(\mathrm{Prime}(m) \wedge n = \sum_{i<m} c_{\mathrm{Prime}}(i)), \end{aligned}$$

so $p_0, p_1, p_2, \ldots$ gives the enumeration of primes in increasing order. The estimate $p_n \leq 2^{2^n}$ for the nth prime p_n can be proved by induction on n: For $n = 0$ this is clear, and for $n \geq 1$ we obtain

$$p_n \leq p_0 p_1 \cdots p_{n-1} + 1 \leq 2^{2^0} 2^{2^1} \cdots 2^{2^{n-1}} + 1 = 2^{2^n - 1} + 1 < 2^{2^n}.$$

2.2.3. The class $\mathcal{E}$.

DEFINITION. The class $\mathcal{E}$ consists of those number theoretic functions which can be defined from the initial functions: constant 0, successor S, projections (onto the ith coordinate), addition $+$, modified subtraction $\dot{-}$, multiplication $\cdot$ and exponentiation 2^x, by applications of composition and bounded minimization.

The remarks above show immediately that the characteristic functions of the equality and less than relations lie in $\mathcal{E}$, and that (by the proof of the lemma) the relations in $\mathcal{E}$ are closed under propositional connectives and bounded quantifiers.

Furthermore the above examples show that all the functions in the class $\mathcal{E}$ are elementary. We now prove the converse, which will be useful later.

LEMMA. *There are "pairing functions" π, π_1, π_2 in $\mathcal{E}$ with the following properties:*

(a) π *maps* $\mathbb{N} \times \mathbb{N}$ *bijectively onto* $\mathbb{N}$,
(b) $\pi(a, b) + b + 2 \leq (a + b + 1)^2$ *for* $a + b \geq 1$, *hence* $\pi(a, b) < (a + b + 1)^2$,
(c) $\pi_1(c), \pi_2(c) \leq c$,
(d) $\pi(\pi_1(c), \pi_2(c)) = c$,

(e) $\pi_1(\pi(a,b)) = a$,
(f) $\pi_2(\pi(a,b)) = b$.

PROOF. Enumerate the pairs of natural numbers as follows:

$$\begin{array}{cccc} \vdots & & & \\ 6 & \dots & & \\ 3 & 7 & \dots & \\ 1 & 4 & 8 & \dots \\ 0 & 2 & 5 & 9 \quad \dots \end{array}$$

At position $(0,b)$ we clearly have the sum of the lengths of the preceding diagonals, and on the next diagonal $a+b$ remains constant. Let $\pi(a,b)$ be the number written at position (a,b). Then we have

$$\pi(a,b) = \big(\sum_{i\le a+b} i\big) + a = \frac{1}{2}(a+b)(a+b+1) + a.$$

Clearly $\pi\colon \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ is bijective. Moreover, $a,b \le \pi(a,b)$ and in case $\pi(a,b) \neq 0$ also $a < \pi(a,b)$. Let

$$\pi_1(c) := \mu_{x\le c}\exists_{y\le c}(\pi(x,y) = c),$$
$$\pi_2(c) := \mu_{y\le c}\exists_{x\le c}(\pi(x,y) = c).$$

Then clearly $\pi_i(c) \le c$ for $i \in \{1,2\}$ and

$$\pi_1(\pi(a,b)) = a, \quad \pi_2(\pi(a,b)) = b, \quad \pi(\pi_1(c),\pi_2(c)) = c.$$

π, π_1 and π_2 are in $\mathcal{E}$ by definition. For $\pi(a,b)$ we have the estimate

$$\pi(a,b) + b + 2 \le (a+b+1)^2 \quad \text{for } a+b \ge 1.$$

This follows with $n := a+b$ from

$$\frac{1}{2}n(n+1) + n + 2 \le (n+1)^2 \quad \text{for } n \ge 1,$$

which is equivalent to $n(n+1) + 2(n+1) \le 2((n+1)^2 - 1)$ and hence to $(n+2)(n+1) \le 2n(n+2)$, which holds for $n \ge 1$. □

The proof shows that π, π_1 and π_2 are in fact subelementary.

THEOREM (Gödel's β-function). *There is in $\mathcal{E}$ a function β with the following property: For every sequence $a_0,\dots,a_{n-1} < b$ of numbers less than b we can find a number $c \le 4\cdot 4^{n(b+n+1)^4}$ such that $\beta(c,i) = a_i$ for all $i < n$.*

PROOF. Let

$$a := \pi(b,n) \quad \text{and} \quad d := \prod_{i<n}\big(1+\pi(a_i,i)a!\big).$$

From $a!$ and d we can, for each given $i < n$, reconstruct the number a_i as the unique $x < b$ such that

$$1+\pi(x,i)a! \mid d.$$

For clearly a_i is such an x, and if some $x < b$ were to satisfy the same condition, then because $\pi(x,i) < a$ and the numbers $1+ka!$ are relatively prime for $k \le a$, we would have $\pi(x,i) = \pi(a_j,j)$ for some $j < n$. Hence $x = a_j$ and $i = j$, thus $x = a_i$. – Therefore

$$a_i = \mu_{x<b}\exists_{z<d}((1+\pi(x,i)a!)z = d).$$

We can now define Gödel's β-function as

$$\beta(c,i) := \mu_{x<\pi_1(c)}\exists_{z<\pi_2(c)}((1+\pi(x,i)\cdot\pi_1(c))\cdot z = \pi_2(c)).$$

Clearly β is in $\mathcal{E}$. Furthermore with $c := \pi(a!,d)$ we see that $\beta(c,i) = a_i$. It is then not difficult to estimate the given bound on c, using $\pi(b,n) < (b+n+1)^2$. □

The above definition of β shows that it is subelementary.

2.2.4. Closure properties of $\mathcal{E}$.

THEOREM. *The class $\mathcal{E}$ is closed under limited recursion. Thus if g, h, k are given functions in $\mathcal{E}$ and f is defined from them according to the schema*

$$\begin{aligned} f(\vec{m},0) &= g(\vec{m}),\\ f(\vec{m},n+1) &= h(n,f(\vec{m},n),\vec{m}),\\ f(\vec{m},n) &\le k(\vec{m},n), \end{aligned}$$

then f is in $\mathcal{E}$ also.

PROOF. Let f be defined from g, h and k in $\mathcal{E}$, by limited recursion as above. Using Gödel's β-function as in the last theorem we can find for any given $\vec{m}, n$ a number c such that $\beta(c,i) = f(\vec{m},i)$ for all $i \le n$. Let $R(\vec{m},n,c)$ be the relation

$$\beta(c,0) = g(\vec{m}) \wedge \forall_{i<n}(\beta(c,i+1) = h(i,\beta(c,i),\vec{m}))$$

and note by the remarks above that its characteristic function is in $\mathcal{E}$. It is clear, by induction, that if $R(\vec{m},n,c)$ holds then $\beta(c,i) = f(\vec{m},i)$, for all $i \le n$. Therefore we can define f explicitly by the equation

$$f(\vec{m},n) = \beta(\mu_c R(\vec{m},n,c),n).$$

f will lie in $\mathcal{E}$ if μ_c can be bounded by an $\mathcal{E}$ function. However, the theorem on Gödel's β-function gives a bound $4 \cdot 4^{(n+1)(b+n+2)^4}$, where in this case b can be taken as the maximum of $k(\vec{m}, i)$ for $i \leq n$. But this can be defined in $\mathcal{E}$ as $k(\vec{m}, i_0)$, where $i_0 = \mu_{i \leq n} \forall_{j \leq n}(k(\vec{m}, j) \leq k(\vec{m}, i))$. Hence μ_c can be bounded by an $\mathcal{E}$ function. □

REMARK. Note that it is in this proof only that the exponential function is required, in providing a bound for μ.

COROLLARY. *$\mathcal{E}$ is the class of all elementary functions.*

PROOF. It is sufficient merely to show that $\mathcal{E}$ is closed under bounded sums and bounded products. Suppose for instance, that f is defined from g in $\mathcal{E}$ by bounded summation: $f(\vec{m}, n) = \sum_{i<n} g(\vec{m}, i)$. Then f can be defined by limited recursion, as follows

$$\begin{aligned} f(\vec{m}, 0) &= 0 \\ f(\vec{m}, n+1) &= f(\vec{m}, n) + g(\vec{m}, n) \\ f(\vec{m}, n) &\leq n \cdot \max_{i<n} g(\vec{m}, i) \end{aligned}$$

and the functions (including the bound) from which it is defined are in $\mathcal{E}$. Thus f is in $\mathcal{E}$ by the theorem. If instead, f is defined by bounded product, then proceed similarly. □

2.2.5. Coding finite lists. Computation on lists is a practical necessity, so because we are basing everything here on the single data type $\mathbb{N}$ we must develop some means of "coding" finite lists or sequences of natural numbers into $\mathbb{N}$ itself. There are various ways to do this and we shall adopt one of the most traditional, based on the pairing functions π, π_1, π_2.

The empty sequence is coded by the number 0 and a sequence n_0, n_1, $\ldots$, n_{k-1} is coded by the "sequence number"

$$\langle n_0, n_1, \ldots, n_{k-1} \rangle = \pi'(\ldots \pi'(\pi'(0, n_0), n_1), \ldots, n_{k-1})$$

with $\pi'(a, b) := \pi(a, b) + 1$, thus recursively,

$$\begin{aligned} \langle\rangle &:= 0, \\ \langle n_0, n_1, \ldots, n_k \rangle &:= \pi'(\langle n_0, n_1, \ldots, n_{k-1} \rangle, n_k). \end{aligned}$$

Because of the surjectivity of π, every number a can be decoded uniquely as a sequence number $a = \langle n_0, n_1, \ldots, n_{k-1} \rangle$. If a is greater than zero, $\mathrm{hd}(a) := \pi_2(a \dot{-} 1)$ is the "head" (i.e., rightmost element) and $\mathrm{tl}(a) := \pi_1(a \dot{-} 1)$ is the "tail" of the list. The kth iterate of tl is denoted $\mathrm{tl}^{(k)}$ and since $\mathrm{tl}(a)$ is less than or equal to a, $\mathrm{tl}^{(k)}(a)$ is elementarily definable (by limited recursion).

Thus we can define elementarily the "length" and "decoding" functions:

$$\begin{aligned}\mathrm{lh}(a) &:= \mu_{k\le a}(\mathrm{tl}^{(k)}(a) = 0),\\ (a)_i &:= \mathrm{hd}(\mathrm{tl}^{(\mathrm{lh}(a)\dot{-}(i+1))}(a)).\end{aligned}$$

Then if $a = \langle n_0, n_1, \ldots, n_{k-1}\rangle$ it is easy to check that

$$\mathrm{lh}(a) = k \text{ and } (a)_i = n_i \text{ for each } i < k.$$

Furthermore $(a)_i = 0$ when $i \ge \mathrm{lh}(a)$. We shall write $(a)_{i,j}$ for $((a)_i)_j$ and $(a)_{i,j,k}$ for $(((a)_i)_j)_k$. This elementary coding machinery will be used at various crucial points in the following.

Note that our previous remarks show that the functions $\mathrm{lh}(\cdot)$ and $(a)_i$ are subelementary, and so is $\langle n_0, n_1, \ldots, n_{k-1}\rangle$ for each fixed k.

Lemma (Estimate for sequence numbers).

$$(n+1)k \le \langle \underbrace{n, \ldots, n}_{k}\rangle < (n+1)^{2^k}.$$

Proof. We prove a slightly strengthened form of the second estimate:

$$\langle \underbrace{n, \ldots, n}_{k}\rangle + n + 1 \le (n+1)^{2^k},$$

by induction on k. For $k = 0$ the claim is clear. In the step $k \mapsto k+1$ we have

$$\begin{aligned}\langle \underbrace{n, \ldots, n}_{k+1}\rangle + n + 1 &= \pi(\langle \underbrace{n, \ldots, n}_{k}\rangle, n) + n + 2\\ &\le (\langle \underbrace{n, \ldots, n}_{k}\rangle + n + 1)^2 \quad \text{by the lemma in Section 2.2.3}\\ &\le (n+1)^{2^{k+1}} \quad \text{by induction hypothesis.}\end{aligned}$$

For the first estimate the base case $k = 0$ is clear, and in the step we have

$$\begin{aligned}\langle \underbrace{n, \ldots, n}_{k+1}\rangle &= \pi(\langle \underbrace{n, \ldots, n}_{k}\rangle, n) + 1\\ &\ge \langle \underbrace{n, \ldots, n}_{k}\rangle + n + 1\\ &\ge (n+1)(k+1) \quad \text{by induction hypothesis.}\end{aligned}$$

□

Concatenation of sequence numbers $b * a$ is defined thus:

$$\begin{aligned}b * \langle\rangle &:= b,\\ b * \langle n_0, n_1, \ldots, n_k\rangle &:= \pi(b * \langle n_0, n_1, \ldots, n_{k-1}\rangle, n_k) + 1.\end{aligned}$$

To check that this operation is also elementary, define $h(b, a, i)$ by recursion on i as follows.

$$\begin{aligned} h(b, a, 0) &= b, \\ h(b, a, i+1) &= \pi(h(b, a, i), (a)_i) + 1 \end{aligned}$$

and note that since

$$h(b, a, i) = \langle (b)_0, \ldots, (b)_{\mathrm{lh}(b) \dot{-} 1}, (a)_0, \ldots, (a)_{i \dot{-} 1} \rangle \quad \text{for } i \leq \mathrm{lh}(a)$$

it follows from the estimate above that $h(a, b, i) \leq (b + a)^{2^{\mathrm{lh}(b)+i}}$. Thus h is definable by limited recursion from elementary functions and hence is itself elementary. Finally

$$b * a = h(b, a, \mathrm{lh}(a)).$$

2.3. Kleene's normal form theorem

2.3.1. Program numbers. The three types of register machine instructions I can be coded by "instruction numbers" $\sharp I$ thus, where v_0, v_1, v_2, ... is a list of all variables used to denote registers:

If I is "$v_j := 0$" then $\sharp I = \langle 0, j \rangle$.
If I is "$v_j := v_j + 1$" then $\sharp I = \langle 1, j \rangle$.
If I is "**if** $v_j = v_l$ **then** I_m **else** I_n" then $\sharp I = \langle 2, j, l, m, n \rangle$.

Clearly, using the sequence coding and decoding apparatus above, we can check elementarily whether or not a given number is an instruction number.

Any register machine program $P = I_0, I_1, \ldots, I_{k-1}$ can then be coded by a "program number" or "index" $\sharp P$ thus:

$$\sharp P = \langle \sharp I_0, \sharp I_1, \ldots, \sharp I_{k-1} \rangle$$

and again (although it is tedious) we can elementarily check whether or not a given number is indeed of the form $\sharp P$ for some program P. Tradition has it that e is normally reserved as a variable over putative program numbers.

Standard program constructs such as those in Section 2.1 have associated "index-constructors", i.e., functions which, given indices of the subprograms, produce an index for the constructed program. The point is that for standard program constructs the associated index-constructor functions are elementary. For example, there is an elementary index-constructor comp such that, given programs P_0, P_1 with indices e_0, e_1, $\mathrm{comp}(e_0, e_1)$ is an index of the program P_0 **;** P_1. A moment's thought should convince the reader that the appropriate definition of comp is as follows:

$$\mathrm{comp}(e_0, e_1) = e_0 * \langle r(e_0, e_1, 0), r(e_0, e_1, 1), \ldots, r(e_0, e_1, \mathrm{lh}(e_1) \dot{-} 1) \rangle$$

where $r(e_0, e_1, i) =$

$$\begin{cases} \langle 2, (e_1)_{i,1}, (e_1)_{i,2}, (e_1)_{i,3} + \mathrm{lh}(e_0), (e_1)_{i,4} + \mathrm{lh}(e_0)\rangle & \text{if } (e_1)_{i,0} = 2 \\ (e_1)_i & \text{otherwise} \end{cases}$$

re-addresses the jump instructions in P_1. Clearly r and hence comp are elementary functions.

DEFINITION. Henceforth, $\varphi_e^{(r)}$ denotes the partial function computed by the register machine program with program number e, operating on the input registers $v_1, \ldots, v_r$ and with output register v_0. There is no loss of generality here, since the variables in any program can always be renamed so that $v_1, \ldots, v_r$ become the input registers and v_0 the output. If e is not a program number, or it is but does not operate on the right variables, then we adopt the convention that $\varphi_e^{(r)}(n_1, \ldots, n_r)$ is undefined for all inputs $n_1, \ldots, n_r$. Alternative notation for $\varphi_e^{(r)}(n_1, \ldots, n_r)$ is $\{e\}(n_1, \ldots, n_r)$.

2.3.2. Normal form.

THEOREM (Kleene's normal form). *For each arity r there is an elementary function U and an elementary relation T such that, for all e and all inputs $n_1, \ldots, n_r$,*

(a) $\varphi_e^{(r)}(n_1, \ldots, n_r)$ *is defined if and only if* $\exists_s T(e, n_1, \ldots, n_r, s)$,

(b) $\varphi_e^{(r)}(n_1, \ldots, n_r) = U(e, n_1, \ldots, n_r, \mu_s T(e, n_1, \ldots, n_r, s))$.

PROOF. A computation of a register machine program $P(v_1, \ldots, v_r; v_0)$ on numerical inputs $\vec{n} = n_1, \ldots, n_r$ proceeds deterministically, step by step, each step corresponding to the execution of one instruction. Let e be its program number, and let $v_0, \ldots, v_l$ be all the registers used by P, including the "working registers", so $r \leq l$.

The "state" of the computation at step s is defined to be the sequence number

$$\mathrm{state}(e, \vec{n}, s) = \langle e, i, m_0, m_1, \ldots, m_l\rangle$$

where $m_0, m_1, \ldots, m_l$ are the values stored in the registers $v_0, v_1, \ldots, v_l$ after step s is completed, and the next instruction to be performed is the i-th one, thus $(e)_i$ is its instruction number.

The "state transition function" $\mathrm{tr}\colon \mathbb{N} \to \mathbb{N}$ computes the "next state". So suppose that $x = \langle e, i, m_0, m_1, \ldots, m_l\rangle$ is any putative state. Then in what follows, $e = (x)_0$, $i = (x)_1$, and $m_j = (x)_{j+2}$ for each $j \leq l$. The definition of $\mathrm{tr}(x)$ is therefore as follows:

$$\mathrm{tr}(x) = \langle e, i', m_0', m_1', \ldots, m_l'\rangle$$

where

(i) If $(e)_i = \langle 0, j\rangle$ where $j \leq l$ then $i' = i+1$, $m'_j = 0$, and all other registers remain unchanged, i.e., $m'_k = m_k$ for $k \neq j$.
(ii) If $(e)_i = \langle 1, j\rangle$ where $j \leq l$ then $i' = i+1$, $m'_j = m_j + 1$, and all other registers remain unchanged.
(iii) If $(e)_i = \langle 2, j_0, j_1, i_0, i_1\rangle$ where $j_0, j_1 \leq l$ and $i_0, i_1 \leq \mathrm{lh}(e)$ then $i' = i_0$ or $i' = i_1$ according as $m_{j_0} = m_{j_1}$ or not, and all registers remain unchanged, i.e., $m'_j = m_j$ for all $j \leq l$.
(iv) Otherwise, if e is not a program number, or if it refers to a register v_k with $l < k$, or if $\mathrm{lh}(e) \leq i$, then $\mathrm{tr}(x)$ simply repeats the same state x so $i' = i$, and $m'_j = m_j$ for every $j \leq l$.

Clearly tr is an *elementary* function, since it is defined by elementarily decidable cases, with (a great deal of) elementary decoding and re-coding involved in each case.

Consequently, the "state function" $\mathrm{state}(e, \vec{n}, s)$ is also *elementary* because it can be defined by iterating the transition function by limited recursion on s as follows:

$$\begin{aligned}
\mathrm{state}(e, \vec{n}, 0) &= \langle e, 0, 0, n_1, \ldots, n_r, 0, \ldots, 0\rangle \\
\mathrm{state}(e, \vec{n}, s+1) &= \mathrm{tr}(\mathrm{state}(e, \vec{n}, s)) \\
\mathrm{state}(e, \vec{n}, s) &\leq h(e, \vec{n}, s)
\end{aligned}$$

where for the bounding function h we can take

$$h(e, \vec{n}, s) = \langle e, e\rangle * \langle \max(\vec{n}\,) + s, \ldots, \max(\vec{n}\,) + s\rangle.$$

This is because the maximum value of any register at step s cannot be greater than $\max(\vec{n}\,) + s$. Now this expression clearly is elementary, since $\langle m, \ldots, m\rangle$ with i occurrences of m is definable by a limited recursion with bound $(m+1)^{2^i}$, by the estimate Lemma in Section 2.2.5.

Now recall that if program P has program number e then computation terminates when instruction $I_{\mathrm{lh}(e)}$ is encountered. Thus we can define the "termination relation" $T(e, \vec{n}, s)$, meaning "computation terminates at step s", by

$$T(e, \vec{n}, s) := ((\mathrm{state}(e, \vec{n}, s))_1 = \mathrm{lh}(e)).$$

Clearly T is elementary and

$$\varphi_e^{(r)}(\vec{n}\,) \text{ is defined} \leftrightarrow \exists_s T(e, \vec{n}, s).$$

The output on termination is the value of register v_0, so if we define the "output function" $U(e, \vec{n}, s)$ by

$$U(e, \vec{n}, s) := (\mathrm{state}(e, \vec{n}, s))_2$$

then U is also elementary and

$$\varphi_e^{(r)}(\vec{n}\,) = U(e, \vec{n}, \mu_s T(e, \vec{n}, s)). \qquad \square$$

2.3.3. Σ^0_1-definable relations and μ-recursive functions. A relation R of arity r is said to be Σ^0_1-*definable* if there is an elementary relation E, say of arity $r+l$, such that for all $\vec{n} = n_1, \ldots, n_r$,

$$R(\vec{n}\,) \leftrightarrow \exists_{k_1,\ldots k_l} E(\vec{n}, k_1, \ldots, k_l).$$

A partial function φ is said to be Σ^0_1-*definable* if its graph

$$\{\, (\vec{n}, m) \mid \varphi(\vec{n}\,) \text{ is defined and } = m \,\}$$

is Σ^0_1-definable.

To say that a non-empty relation R is Σ^0_1-definable is equivalent to saying that the set of all sequences $\langle \vec{n} \rangle$ satisfying R can be enumerated (possibly with repetitions) by some elementary function $f\colon \mathbb{N} \to \mathbb{N}$. Such relations are called *elementarily enumerable*. For choose any fixed sequence $\langle a_1, \ldots, a_r \rangle$ satisfying R and define

$$f(m) = \begin{cases} \langle (m)_1, \ldots, (m)_r \rangle & \text{if } E((m)_1, \ldots, (m)_{r+l}) \\ \langle a_1, \ldots, a_r \rangle & \text{otherwise.} \end{cases}$$

Conversely if R is elementarily enumerated by f then

$$R(\vec{n}\,) \leftrightarrow \exists_m (f(m) = \langle \vec{n} \rangle)$$

is a Σ^0_1-definition of R.

The *μ-recursive functions* are those (partial) functions which can be defined from the initial functions: constant 0, successor S, projections (onto the i-th coordinate), addition $+$, modified subtraction $\dot{-}$ and multiplication $\cdot$, by applications of composition and unbounded minimization. Note that it is through unbounded minimization that partial functions may arise.

LEMMA. *Every elementary function is μ-recursive.*

PROOF. By simply removing the bounds on μ in the lemmas in Section 2.2.3 one obtains μ-recursive definitions of the pairing functions π, π_1, π_2 and of Gödel's β-function. Then by removing all mention of bounds from the theorem in Section 2.2.4 one sees that the μ-recursive functions are closed under (unlimited) primitive recursive definitions: $f(\vec{m}, 0) = g(\vec{m})$, $f(\vec{m}, n+1) = h(n, f(\vec{m}, n), \vec{m})$. Thus one can μ-recursively define bounded sums and bounded products, and hence all elementary functions. □

2.3.4. Computable functions.

DEFINITION. The *while programs* are those programs which can be built up from assignment statements $x := 0$, $x := y$, $x := y + 1$, $x := y \dot{-} 1$, by conditionals, composition, for-loops and while-loops as in Section 2.1 (on program constructs).

THEOREM. *The following are equivalent:*

(a) φ *is register machine computable,*
(b) φ *is* Σ_1^0*-definable,*
(c) φ *is* μ*-recursive,*
(d) φ *is computable by a while program.*

PROOF. The normal form theorem shows immediately that every register machine computable function $\varphi_e^{(r)}$ is Σ_1^0-definable since

$$\varphi_e^{(r)}(\vec{n}\,) = m \leftrightarrow \exists_s(T(e,\vec{n},s) \wedge U(e,\vec{n},s) = m)$$

and the relation $T(e,\vec{n},s) \wedge U(e,\vec{n},s) = m$ is clearly elementary. If φ is Σ_1^0-definable, say

$$\varphi(\vec{n}\,) = m \leftrightarrow \exists_{k_1,\ldots k_l} E(\vec{n}, m, k_1, \ldots, k_l),$$

then φ can be defined μ-recursively by

$$\varphi(\vec{n}\,) = (\mu_m E(\vec{n}, (m)_0, (m)_1, \ldots, (m)_l)\,)_0\ ,$$

using the fact (above) that elementary functions are μ-recursive. The examples of computable functionals in Section 2.1 show how the definition of any μ-recursive function translates automatically into a while program. Finally, Section 2.1 shows how to implement any while program on a register machine. □

Henceforth *computable* means "register machine computable" or any of its equivalents.

COROLLARY. *The function* $\varphi_e^{(r)}(n_1, \ldots, n_r)$ *is a computable partial function of the* $r+1$ *variables* $e, n_1, \ldots, n_r$.

PROOF. Immediate from the normal form. □

LEMMA. *Let* R *and* $\bar{R}$ *be disjoint inhabited relations with* $\forall_{\vec{n}}(R\vec{n} \vee \bar{R}\vec{n})$. *Then* R *is computable if and only if both* R *and* $\bar{R}$ *are* Σ_1^0*-definable.*

PROOF. We assume (for simplicity) that R and $\bar{R}$ are unary.

"$\rightarrow$". By the theorem above every computable relation is Σ_1^0-definable, and with R clearly $\bar{R}$ is computable.

"$\leftarrow$". Let $f, g \in \mathcal{E}$ enumerate R and $\bar{R}$, respectively. Then

$$h(n) := \mu_i(f(i) = n \vee g(i) = n)$$

is a total μ-recursive function, and $R(n) \leftrightarrow f(h(n)) = n$. □

2.3.5. Undecidability of the halting problem. The above corollary says that there is a single "universal" program which, given numbers e and $\vec{n}$, computes $\varphi_e^{(r)}(\vec{n}\,)$ if it is defined. However, we cannot decide in advance whether or not it will be defined. There is no program which, given e and $\vec{n}$, computes the total function

$$h(e,\vec{n}\,) = \begin{cases} 1 & \text{if } \varphi_e^{(r)}(\vec{n}\,) \text{ is defined} \\ 0 & \text{if } \varphi_e^{(r)}(\vec{n}\,) \text{ is undefined.} \end{cases}$$

For suppose there were such a program. Then the function

$$\psi(\vec{n}\,) = \mu_m(h(n_1,\vec{n}\,) = 0)$$

would be computable, say with fixed program number e_0, and therefore

$$\varphi_{e_0}^{(r)}(\vec{n}\,) = \begin{cases} 0 & \text{if } h(n_1,\vec{n}\,) = 0 \\ \text{undefined} & \text{if } h(n_1,\vec{n}\,) = 1. \end{cases}$$

But then fixing $n_1 = e_0$ gives

$$\varphi_{e_0}^{(r)}(\vec{n}\,) \text{ defined} \leftrightarrow h(e_0,\vec{n}\,) = 0 \leftrightarrow \varphi_{e_0}^{(r)}(\vec{n}\,) \text{ undefined},$$

a contradiction. Hence the relation $R(e,\vec{n}\,)$, which holds if and only if $\varphi_e^{(r)}(\vec{n}\,)$ is defined, is not recursive. It is however Σ_1^0-definable.

CHAPTER 3

Gödel's Theorems

We now bring proof and recursion together. A principal object of study in this chapter are the elementary functions, which are adequate for the arithmetization of syntax leading to Gödel's two incompleteness theorems.

3.1. The notion of truth in formal theories

We consider the question whether there is a truth formula $B(z)$ such that in appropriate theories T we have $T \vdash A \leftrightarrow B(\underline{\ulcorner A \urcorner})$ for all sentences A. Here $\ulcorner A \urcorner$ is the "Gödel number" of A, and $\underline{a}$ is the "numeral" denoting $a \in \mathbb{N}$; both notions are defined in Section 3.1.1 below. The result will be that this is impossible, under rather weak assumptions on the theory T. Technically, the issue will be to have a syntactic substitute of the notion of definability by "representability" within a formal theory. This notion is defined in Section 3.1.2.

3.1.1. Gödel numbers. We will assign numbers – so-called Gödel numbers, GN for short – to the syntactical constructs developed in Chapter 1: terms, formulas and derivations. Using the elementary sequence-coding and decoding machinery developed earlier we will be able to construct the code number of a composed object from its parts, and conversely to disassemble the code number of a composed object into the code numbers of its parts.

Let $\mathcal{L}$ be a countable first-order language. Assume that we have injectively assigned to every n-ary relation symbol R a *symbol number* $\mathrm{sn}(R)$ of the form $\langle 1, n, i\rangle$ and to every n-ary function symbol f a symbol number $\mathrm{sn}(f)$ of the form $\langle 2, n, j\rangle$. Call $\mathcal{L}$ *elementarily presented* if the set $\mathrm{Symb}_{\mathcal{L}}$ of all these symbol numbers is elementary. In what follows we shall always assume that the languages $\mathcal{L}$ considered are elementarily presented. In particular this applies to every language with finitely many relation and function symbols.

Let $\mathrm{sn}(\mathrm{Var}) := \langle 0\rangle$. For every $\mathcal{L}$-term t we define recursively its Gödel number $\ulcorner t\urcorner$ by

$$\begin{aligned}\ulcorner x_i\urcorner &:= \langle \mathrm{sn}(\mathrm{Var}), i\rangle,\\ \ulcorner f t_1 \dots t_n\urcorner &:= \langle \mathrm{sn}(f), \ulcorner t_1\urcorner, \dots, \ulcorner t_n\urcorner\rangle.\end{aligned}$$

Assign numbers to the logical symbols by $\mathrm{sn}(\to) := \langle 3,0\rangle$ and $\mathrm{sn}(\forall) := \langle 3,1\rangle$. For simplicity we leave out the logical connective $\wedge$ here; it could be treated similarly. We define for every $\mathcal{L}$-formula A its Gödel number $\ulcorner A\urcorner$ by

$$\begin{aligned}\ulcorner R t_1 \dots t_n\urcorner &:= \langle \mathrm{sn}(R), \ulcorner t_1\urcorner, \dots, \ulcorner t_n\urcorner\rangle,\\ \ulcorner A \to B\urcorner &:= \langle \mathrm{sn}(\to), \ulcorner A\urcorner, \ulcorner B\urcorner\rangle,\\ \ulcorner \forall_{x_i} A\urcorner &:= \langle \mathrm{sn}(\forall), i, \ulcorner A\urcorner\rangle.\end{aligned}$$

Assume that 0 is a constant and S is a unary function symbol in $\mathcal{L}$. For every $a \in \mathbb{N}$ the *numeral* $\underline{a} \in \mathrm{Ter}_{\mathcal{L}}$ is defined by $\underline{0} := 0$ and $\underline{n+1} := S\underline{n}$. We can define an elementary function s such that for every formula $C = C(z)$ with $z := x_0$,

$$s(\ulcorner C\urcorner, k) = \ulcorner C(\underline{k})\urcorner;$$

the proof is an exercise.

We define symbol numbers for the names of the natural deduction rules: $\mathrm{sn}(\mathrm{AssVar}) := \langle 4,0\rangle$, $\mathrm{sn}(\to^+) := \langle 4,1\rangle$, $\mathrm{sn}(\to^-) := \langle 4,2\rangle$, $\mathrm{sn}(\forall^+) := \langle 4,3\rangle$, $\mathrm{sn}(\forall^-) := \langle 4,4\rangle$. For a derivation M we define its Gödel number $\ulcorner M\urcorner$ by

$$\begin{aligned}\ulcorner u_i^A\urcorner &:= \langle \mathrm{sn}(\mathrm{AssVar}), i, \ulcorner A\urcorner\rangle,\\ \ulcorner \lambda_{u_i^A} M\urcorner &:= \langle \mathrm{sn}(\to^+), i, \ulcorner A\urcorner, \ulcorner M\urcorner\rangle,\\ \ulcorner M N\urcorner &:= \langle \mathrm{sn}(\to^-), \ulcorner M\urcorner, \ulcorner N\urcorner\rangle,\\ \ulcorner \lambda_{x_i} M\urcorner &:= \langle \mathrm{sn}(\forall^+), i, \ulcorner M\urcorner\rangle,\\ \ulcorner M t\urcorner &:= \langle \mathrm{sn}(\forall^-), \ulcorner M\urcorner, \ulcorner t\urcorner\rangle.\end{aligned}$$

Let T be an $\mathcal{L}$-theory determined by an elementary axiom system Ax_T (containing $\mathrm{Stab}_{\mathcal{L}}$) plus the equality axioms $\mathrm{Eq}_{\mathcal{L}}$:

$$\begin{aligned}&x = x \quad \text{(Reflexivity)},\\ &x = y \to y = x \quad \text{(Symmetry)},\\ &x = y \to y = z \to x = z \quad \text{(Transitivity)},\\ &x_1 = y_1 \to \dots \to x_n = y_n \to f(x_1, \dots, x_n) = f(y_1, \dots, y_n),\\ &x_1 = y_1 \to \dots \to x_n = y_n \to R(x_1, \dots, x_n) \to R(y_1, \dots, y_n),\end{aligned}$$

for all n-ary function symbols f and relation symbols R of the language $\mathcal{L}$. For such axiomatized theories we can define an elementary binary relation Prf_T such that $\mathrm{Prf}_T(d,a)$ holds if and only if d is the GN of a derivation

with a closed end formula with GN a from a context composed of equality axioms and formulas from Ax_T.

3.1.2. Representable relations and functions. In this section we assume that $\mathcal{L}$ is an elementarily presented language with 0, S and $=$ in $\mathcal{L}$, and T an $\mathcal{L}$-theory containing the equality axioms $\mathrm{Eq}_{\mathcal{L}}$.

Definition. A relation $R \subseteq \mathbb{N}^n$ is *representable* in T if there is a formula $A(x_1, \dots, x_n)$ such that

$$\begin{aligned} T &\vdash A(\underline{a_1}, \dots, \underline{a_n}) \quad \text{if } (a_1, \dots, a_n) \in R, \\ T &\vdash \neg A(\underline{a_1}, \dots, \underline{a_n}) \quad \text{if } (a_1, \dots, a_n) \notin R. \end{aligned}$$

A function $f\colon \mathbb{N}^n \to \mathbb{N}$ is called *representable* in T if there is a formula $A(x_1, \dots, x_n, y)$ representing the graph $G_f \subseteq \mathbb{N}^{n+1}$ of f, i.e., such that

$$T \vdash A(\underline{a_1}, \dots, \underline{a_n}, \underline{f(a_1, \dots, a_n)}), \tag{15}$$

$$T \vdash \neg A(\underline{a_1}, \dots, \underline{a_n}, \underline{c}) \quad \text{if } c \neq f(a_1, \dots, a_n) \tag{16}$$

and such that in addition

$$T \vdash A(\underline{a_1}, \dots, \underline{a_n}, y) \to A(\underline{a_1}, \dots, \underline{a_n}, z) \to y{=}z \text{ for all } a_1, \dots, a_n \in \mathbb{N}. \tag{17}$$

Note that in case $T \vdash \underline{b} \neq \underline{c}$ for $b < c$ condition (16) follows from (15) and (17).

Lemma. *If the characteristic function c_R of a relation $R \subseteq \mathbb{N}^n$ is representable in T, then so is the relation R itself.*

Proof. For simplicity assume $n = 1$. Let $A(x, y)$ be a formula representing c_R. We show that $A(x, \underline{1})$ represents the relation R. Assume $a \in R$. Then $c_R(a) = 1$, hence $(a, 1) \in G_{c_R}$, hence $T \vdash A(\underline{a}, \underline{1})$. Conversely, assume $a \notin R$. Then $c_R(a) = 0$, hence $(a, 1) \notin G_{c_R}$, hence $T \vdash \neg A(\underline{a}, \underline{1})$. □

3.1.3. Undefinability of the notion of truth in formal theories.

Lemma (Fixed point lemma). *Assume that all elementary functions are representable in T. Then for every formula $B(z)$ we can find a closed formula A such that*

$$T \vdash A \leftrightarrow B(\underline{\ulcorner A \urcorner}).$$

Proof. Let s be the elementary function introduced in Section 3.1.1 and $A_s(x_1, x_2, x_3)$ a formula representing s in T. Let

$$C(z) := \forall_x (A_s(z, z, x) \to B(x)), \quad A := C(\underline{\ulcorner C \urcorner}),$$

and therefore

$$A = \forall_x (A_s(\underline{\ulcorner C \urcorner}, \underline{\ulcorner C \urcorner}, x) \to B(x)).$$

Because of $s(\ulcorner C\urcorner, \ulcorner C\urcorner) = \ulcorner C(\underline{\ulcorner C\urcorner})\urcorner = \ulcorner A\urcorner$ we can prove in T

$$A_s(\underline{\ulcorner C\urcorner}, \underline{\ulcorner C\urcorner}, x) \leftrightarrow x = \underline{\ulcorner A\urcorner},$$

hence by definition of A also

$$A \leftrightarrow \forall_x(x = \underline{\ulcorner A\urcorner} \to B(x))$$

and therefore

$$A \leftrightarrow B(\underline{\ulcorner A\urcorner}). \qquad \square$$

THEOREM. *Let T be a consistent theory such that all elementary functions are representable in T. Then there cannot exist a formula $B(z)$ defining the notion of truth, i.e., such that for all closed formulas A*

$$T \vdash A \leftrightarrow B(\underline{\ulcorner A\urcorner}).$$

PROOF. Assume we would have such a $B(z)$. Consider the formula $\neg B(z)$ and choose by the fixed point lemma a closed formula A such that

$$T \vdash A \leftrightarrow \neg B(\underline{\ulcorner A\urcorner}).$$

For this A we obtain $T \vdash A \leftrightarrow \neg A$, contradicting the consistency of T. $\square$

3.2. Undecidability and incompleteness

Consider a consistent formal theory T with the property that all recursive functions are representable in T. This is a very weak assumption, as we shall show in the next section: it is always satisfied if the theory allows to develop a certain minimum of arithmetic. We shall show that such a theory necessarily is undecidable. Then we prove Gödel's (first) incompleteness theorem saying that every axiomatized such theory must be incomplete. In fact, we prove a sharpened form of this theorem due to Gödel and then Rosser, which explicitly provides a closed formula A such that neither A nor $\neg A$ is provable in the theory T.

In this section let $\mathcal{L}$ be an elementarily presented language with 0, S, $=$ in $\mathcal{L}$ and T a theory containing the equality axioms $\mathrm{Eq}_{\mathcal{L}}$. Call a relation *recursive* if its (total) characteristic function is recursive. A set S of formulas is called *recursive* (*elementarily enumerable*), if $\ulcorner S\urcorner := \{\, \ulcorner A\urcorner \mid A \in S \,\}$ is recursive (elementarily enumerable).

THEOREM (Undecidability). *Assume that T is a consistent theory such that all recursive functions are representable in T. Then T is not recursive.*

PROOF. Assume that T is recursive. By assumption there exists a formula $B(z)$ representing $\ulcorner T\urcorner$ in T. Choose by the fixed point lemma a closed formula A such that

$$T \vdash A \leftrightarrow \neg B(\underline{\ulcorner A\urcorner}).$$

We shall prove $(*)$ $T \nvdash A$ and $(**)$ $T \vdash A$; this is the desired contradiction.

Ad $(*)$. Assume $T \vdash A$. Then $A \in T$, hence $\ulcorner A \urcorner \in \ulcorner T \urcorner$, hence $T \vdash B(\underline{\ulcorner A \urcorner})$ (because $B(z)$ represents in T the set $\ulcorner T \urcorner$). By the choice of A it follows that $T \vdash \neg A$, which contradicts the consistency of T.

Ad $(**)$. By $(*)$ we know $T \nvdash A$. Therefore $A \notin T$, hence $\ulcorner A \urcorner \notin \ulcorner T \urcorner$ and therefore $T \vdash \neg B(\underline{\ulcorner A \urcorner})$. By the choice of A it follows that $T \vdash A$. $\square$

THEOREM (Gödel-Rosser). *Let T be axiomatized and consistent. Assume that there is a formula $L(x, y)$ – written $x < y$ – such that*

$$T \vdash \forall_{x<\underline{n}}(x = \underline{0} \vee \cdots \vee x = \underline{n-1}), \tag{18}$$

$$T \vdash \forall_x(x = \underline{0} \vee \cdots \vee x = \underline{n} \vee \underline{n} < x). \tag{19}$$

Assume also that every elementary function is representable in T. Then we can find a closed formula A such that neither A nor $\neg A$ is provable in T.

PROOF. We first define $\mathrm{Refut}_T \subseteq \mathbb{N} \times \mathbb{N}$ by

$$\mathrm{Refut}_T(d, a) := \mathrm{Prf}_T(d, \dot{\neg} a)$$

with $\dot{\neg} a := \langle \mathrm{sn}(\to), a, \mathrm{sn}(\bot) \rangle$. Then Refut_T is elementary and $\mathrm{Refut}_T(d, a)$ holds if and only if d is the GN of a derivation of the negation of a formula with GN a from a context composed of equality axioms and formulas from Ax_T. Let $B_{\mathrm{Prf}_T}(x_1, x_2)$ and $B_{\mathrm{Refut}_T}(x_1, x_2)$ be formulas representing Prf_T and Refut_T, respectively. Choose by the fixed point lemma a closed formula A such that

$$T \vdash A \leftrightarrow \forall_x(B_{\mathrm{Prf}_T}(x, \underline{\ulcorner A \urcorner}) \to \exists_{y<x} B_{\mathrm{Refut}_T}(y, \underline{\ulcorner A \urcorner})).$$

A expresses its own underivability, in the form (due to Rosser): "For every proof of me there is a shorter proof of my negation".

We shall show $(*)$ $T \nvdash A$ and $(**)$ $T \nvdash \neg A$.

Ad $(*)$. Assume $T \vdash A$. Choose n such that

$$\mathrm{Prf}_T(n, \ulcorner A \urcorner).$$

Then we also have

$$\text{not } \mathrm{Refut}_T(m, \ulcorner A \urcorner) \qquad \text{for all } m,$$

since T is consistent. Hence

$$T \vdash B_{\mathrm{Prf}_T}(\underline{n}, \underline{\ulcorner A \urcorner}),$$
$$T \vdash \neg B_{\mathrm{Refut}_T}(\underline{m}, \underline{\ulcorner A \urcorner}) \qquad \text{for all } m.$$

By (18) we can conclude

$$T \vdash B_{\mathrm{Prf}_T}(\underline{n}, \underline{\ulcorner A \urcorner}) \wedge \forall_{y<\underline{n}} \neg B_{\mathrm{Refut}_T}(y, \underline{\ulcorner A \urcorner}).$$

Hence

$$T \vdash \exists_x(B_{\mathrm{Prf}_T}(x, \underline{\ulcorner A \urcorner}) \wedge \forall_{y<x} \neg B_{\mathrm{Refut}_T}(y, \underline{\ulcorner A \urcorner})),$$
$$T \vdash \neg A.$$

This contradicts the assumed consistency of T.

Ad $(**)$. Assume $T \vdash \neg A$. Choose n such that

$$\mathrm{Refut}_T(n, \ulcorner A \urcorner).$$

Then we also have

$$\text{not } \mathrm{Prf}_T(m, \ulcorner A \urcorner) \qquad \text{for all } m,$$

since T is consistent. Hence

$$T \vdash B_{\mathrm{Refut}_T}(\underline{n}, \underline{\ulcorner A \urcorner}),$$
$$T \vdash \neg B_{\mathrm{Prf}_T}(\underline{m}, \underline{\ulcorner A \urcorner}) \qquad \text{for all } m.$$

This implies

$$T \vdash \forall_x(B_{\mathrm{Prf}_T}(x, \underline{\ulcorner A \urcorner}) \to \exists_{y<x} B_{\mathrm{Refut}_T}(y, \underline{\ulcorner A \urcorner})),$$

as can be seen easily by cases on x, using (19). Hence $T \vdash A$. But this again contradicts the assumed consistency of T. □

Finally we formulate a variant of this theorem which does not assume that the theory T talks about numbers only. Call T a *theory with defined natural numbers* if there is a formula $N(x)$ – written Nx – such that $T \vdash N0$ and $T \vdash \forall_{x \in N} N(Sx)$ where $\forall_{x \in N} A$ is short for $\forall_x(Nx \to A)$. Representing a function in such a theory of course means that the free variables in (17) are relativized to N:

$$T \vdash \forall_{y,z \in N}(A(\underline{a_1}, \ldots, \underline{a_n}, y) \to A(\underline{a_1}, \ldots, \underline{a_n}, z) \to y{=}z) \text{ for all } \vec{a} \in \mathbb{N}.$$

THEOREM (Gödel-Rosser). *Assume that T is an axiomatized consistent theory with defined natural numbers, and that there is a formula $L(x, y)$ – written $x < y$ – such that*

$$T \vdash \forall_{x \in N}(x < \underline{n} \to x = \underline{0} \vee \ \cdots \vee \ x = \underline{n-1}),$$
$$T \vdash \forall_{x \in N}(x = \underline{0} \vee \ \cdots \vee \ x = \underline{n} \vee \ \underline{n} < x).$$

Assume also that every elementary function is representable in T. Then one can find a closed formula A such that neither A nor $\neg A$ is provable in T.

PROOF. As for the Gödel-Rosser theorem above; just relativize all quantifiers to N. □

3.3. Representability of recursive functions

We show in this section that already very simple theories have the property that all recursive functions are representable in them; an example is a finitely axiomatized arithmetical theory Q due to Robinson (1950). A consequence will be the (even "essential") undecidability of Q.

3.3.1. Weak arithmetical theories.

THEOREM. *Let $\mathcal{L}$ be an elementarily presented language with 0, S, $=$ in $\mathcal{L}$ and T a consistent theory with defined natural numbers containing the equality axioms $\mathrm{Eq}_{\mathcal{L}}$. Assume that*

(20) $T \vdash S\underline{a} \neq 0$ *for all $a \in \mathbb{N}$,*

(21) $T \vdash S\underline{a} = S\underline{b} \to \underline{a} = \underline{b}$ *for all $a, b \in \mathbb{N}$,*

(22) *the functions $+$ and $\cdot$ are representable in T*

and that there is a formula $L(x, y)$ – written $x < y$ – such that

(23) $T \vdash \forall_{x \in N}(x \not< 0)$,

(24) $T \vdash \forall_{x \in N}(x < S\underline{b} \to x < \underline{b} \vee x = \underline{b})$ *for all $b \in \mathbb{N}$,*

(25) $T \vdash \forall_{x \in N}(x < \underline{b} \vee x = \underline{b} \vee \underline{b} < x)$ *for all $b \in \mathbb{N}$.*

Then every recursive function is representable in T.

PROOF. First note that the formulas $x = y$ and $x < y$ actually do represent in T the equality and the less-than relations, respectively. From (20) and (21) we can see immediately that $T \vdash \underline{a} \neq \underline{b}$ when $a \neq b$. Assume $a \not< b$. We show $T \vdash \underline{a} \not< \underline{b}$ by induction on b. $T \vdash \underline{a} \not< 0$ follows from (23). In the step we have $a \not< Sb$, hence $a \not< b$ and $a \neq b$, hence by induction hypothesis and the representability (above) of the equality relation, $T \vdash \underline{a} \not< \underline{b}$ and $T \vdash \underline{a} \neq \underline{b}$, hence by (24) $T \vdash \underline{a} \not< S\underline{b}$. Now assume $a < b$. Then $T \vdash \underline{a} \neq \underline{b}$ and $T \vdash \underline{b} \not< \underline{a}$, hence by (25) $T \vdash \underline{a} < \underline{b}$.

We now show by induction on the definition of μ-recursive functions that every recursive function is representable in T. Recall (from Section 3.1.2) that the second condition (16) in the definition of representability of a function automatically follows from the other two (and hence need not be checked further). This is because $T \vdash \underline{a} \neq \underline{b}$ for $a \neq b$.

The *initial functions* constant 0, successor and projection (onto the i-th coordinate) are trivially represented by the formulas $0 = y$, $Sx = y$ and $x_i = y$ respectively. Addition and multiplication are represented in T by assumption. Recall that the one remaining initial function of μ-recursiveness is $\dot{-}$, but this is definable from the characteristic function of $<$ by $a \dot{-} b = \mu_i(b + i \geq a) = \mu_i(c_<(b + i, a) = 0)$. We now show that the characteristic function of $<$ is representable in T. (It will then follow that

$\dot{-}$ is representable, once we have shown that the representable functions are closed under μ.) We show that

$$A(x_1, x_2, y) := (x_1 < x_2 \wedge y = 1) \vee (x_1 \not< x_2 \wedge y = 0)$$

represents $c_<$. First notice that $\forall_{y,z \in N}(A(\underline{a_1}, \underline{a_2}, y) \to A(\underline{a_1}, \underline{a_2}, z) \to y = z)$ already follows logically from the equality axiom (by cases on the alternatives of A). Assume $a_1 < a_2$. Then $T \vdash \underline{a_1} < \underline{a_2}$, hence $T \vdash A(\underline{a_1}, \underline{a_2}, 1)$. Now assume $a_1 \not< a_2$. Then $T \vdash \underline{a_1} \not< \underline{a_2}$, hence $T \vdash A(\underline{a_1}, \underline{a_2}, 0)$.

For the *composition* case, suppose f is defined from $h, g_1, \ldots, g_m$ by

$$f(\vec{a}\,) = h(g_1(\vec{a}\,), \ldots, g_m(\vec{a}\,)).$$

By induction hypothesis we already have representing formulas $A_{g_i}(\vec{x}, y_i)$ and $A_h(\vec{y}, z)$. As representing formula for f we take

$$A_f := \exists_{\vec{y} \in N}(A_{g_1}(\vec{x}, y_1) \wedge \cdots \wedge A_{g_m}(\vec{x}, y_m) \wedge A_h(\vec{y}, z)).$$

Assume $f(\vec{a}\,) = c$. Then there are $b_1, \ldots, b_m$ such that $T \vdash A_{g_i}(\underline{\vec{a}}, \underline{b_i})$ for each i, and $T \vdash A_h(\underline{\vec{b}}, \underline{c})$ so by logic $T \vdash A_f(\underline{\vec{a}}, \underline{c})$. It remains to show uniqueness $T \vdash \forall_{z_1, z_2 \in N}(A_f(\underline{\vec{a}}, z_1) \to A_f(\underline{\vec{a}}, z_2) \to z_1 = z_2)$. But this follows by logic from the induction hypothesis for g_i, which gives

$$T \vdash \forall_{y_{1i}, y_{2i} \in N}(A_{g_i}(\underline{\vec{a}}, y_{1i}) \to A_{g_i}(\underline{\vec{a}}, y_{2i}) \to y_{1i} = y_{2i} = \underline{g_i(\vec{a}\,)})$$

and the induction hypothesis for h, which gives

$$T \vdash \forall_{z_1, z_2 \in N}(A_h(\underline{\vec{b}}, z_1) \to A_h(\underline{\vec{b}}, z_2) \to z_1 = z_2) \quad \text{with } b_i = g_i(\vec{a}\,).$$

For the μ case, suppose f is defined from g (taken here to be binary for notational convenience) by $f(a) = \mu_i(g(i, a) = 0)$, assuming $\forall_a \exists_i (g(i, a) = 0)$. By induction hypothesis we have a formula $A_g(y, x, z)$ representing g. In this case we represent f by the formula

$$A_f(x, y) := Ny \wedge A_g(y, x, 0) \wedge \forall_{v \in N}(v < y \to \exists_{u \in N; u \neq 0} A_g(v, x, u)).$$

We first show the representability condition (15), that is $T \vdash A_f(\underline{a}, \underline{b})$ when $f(a) = b$. Because of the form of A_f this follows from the assumed representability of g together with $T \vdash \forall_{v \in N}(v < \underline{b} \to v = \underline{0} \vee \cdots \vee v = \underline{b - 1})$.

We now tackle the uniqueness condition (17). Given a, let $b := f(a)$ (thus $g(b, a) = 0$ and b is the least such). It suffices to show

$$T \vdash \forall_{y \in N}(A_f(\underline{a}, y) \to y = \underline{b}).$$

We prove $T \vdash \forall_{y \in N}(y < \underline{b} \to \neg A_f(\underline{a}, y))$ and $T \vdash \forall_{y \in N}(\underline{b} < y \to \neg A_f(\underline{a}, y))$, and then appeal to the trichotomy law.

We first show $T \vdash \forall_{y \in N}(y < \underline{b} \to \neg A_f(\underline{a}, y))$. Now since, for any $i < b$, $T \vdash \neg A_g(\underline{i}, \underline{a}, 0)$ by the assumed representability of g, we obtain immediately $T \vdash \neg A_f(\underline{a}, \underline{i})$. Hence because of $T \vdash \forall_{y \in N}(y < \underline{b} \to y = \underline{0} \vee \cdots \vee y = \underline{b - 1})$ the claim follows.

Secondly, $T \vdash \forall_{y\in N}(\underline{b} < y \to \neg A_f(\underline{a}, y))$ follows almost immediately from $T \vdash \forall_{y\in N}(\underline{b} < y \to A_f(\underline{a}, y) \to \exists_{u\in N;u\neq 0} A_g(\underline{b}, \underline{a}, u))$ and the uniqueness for g, $T \vdash \forall_{u\in N}(A_g(\underline{b}, \underline{a}, u) \to u = 0)$. □

3.3.2. Robinson's theory Q. We conclude this section by considering a special and particularly simple arithmetical theory due originally to Robinson (1950). Let $\mathcal{L}_1$ be the language given by 0, S, $+$, $\cdot$ and $=$, and let Q be the theory determined by the axioms $\mathrm{Eq}_{\mathcal{L}_1}$ and

$$Sx \neq 0, \tag{26}$$

$$Sx = Sy \to x = y, \tag{27}$$

$$x + 0 = x, \tag{28}$$

$$x + Sy = S(x + y), \tag{29}$$

$$x \cdot 0 = 0, \tag{30}$$

$$x \cdot Sy = x \cdot y + x, \tag{31}$$

$$\exists_z(x + Sz = y) \vee x = y \vee \exists_z(y + Sz = x). \tag{32}$$

THEOREM (Robinson's Q). *Every consistent theory $T \supseteq Q$ fulfills the assumptions of the previous theorem w.r.t. the definition $L(x, y) := \exists_z(x + Sz = y)$ of the $<$-relation. Hence every recursive function is representable in T.*

PROOF. We show that T satisfies the conditions of the previous theorem. For (20) and (21) this is clear. For (22) we can take $x + y = z$ and $x \cdot y = z$ as representing formulas. For (23) we have to show $\neg\exists_z(x + Sz = 0)$; this follows from (29) and (26). For the proof of (24) we need the auxiliary proposition

$$x = 0 \vee \exists_y(x = 0 + Sy), \tag{33}$$

which will be attended to below. Assume $x + Sz = S\underline{b}$, hence also $S(x + z) = S\underline{b}$ and therefore $x + z = \underline{b}$. We must show $\exists_{y'}(x + Sy' = \underline{b}) \vee x = \underline{b}$. But this follows from (33) for z. In case $z = 0$ we obtain $x = \underline{b}$, and in case $\exists_y(z = 0 + Sy)$ we have $\exists_{y'}(x + Sy' = \underline{b})$, since $0 + Sy = S(0 + y)$. Thus (24) is proved. (25) follows immediately from (32). For the proof of (33) we use (32) with $y = 0$. It clearly suffices to exclude the first case $\exists_z(x + Sz = 0)$. But this means $S(x + z) = 0$, contradicting (26). □

COROLLARY (Essential undecidability of Q). *Every consistent theory $T \supseteq Q$ in an elementarily presented language is non-recursive.*

PROOF. This follows from the theorem above and the undecidability theorem in Section 3.2. □

COROLLARY (Undecidability of logic). *The set of formulas derivable in minimal logic is non-recursive.*

PROOF. Otherwise Q would be recursive, because a formula A is derivable in Q if and only if the implication $B \to A$ is derivable, where B is the conjunction of the finitely many axioms and equality axioms of Q. □

REMARK. Note that it suffices that the underlying language contains one binary relation symbol (for $=$), one constant symbol (for 0), one unary function symbol (for S) and two binary functions symbols (for $+$ and $\cdot$). The study of decidable fragments of first-order logic is one of the oldest research areas of mathematical logic. For more information see Börger et al. (1997).

3.3.3. Σ_1-formulas. Reading the above proof of representability, one can see that the representing formulas used are of a restricted form, having no unbounded universal quantifiers and therefore defining Σ^0_1-relations. This will be of crucial importance for our proof of Gödel's second incompleteness theorem to follow, but in addition we need to make a syntactically precise definition of the class of formulas involved, more specific and apparently more restrictive than the notion of Σ_1-formula used earlier. However, as proved in the corollary below, we can still represent all recursive functions even in the weak theory Q by means of Σ_1-formulas in this more restrictive sense. Consequently provable Σ_1-ness will be the same whichever definition we take.

DEFINITION. For the remainder of this chapter, the Σ_1-formulas of the language $\mathcal{L}_1$ will be those generated inductively by the following clauses:

(a) Only atomic formulas of the restricted forms $x = y$, $x \neq y$, $0 = x$, $Sx = y$, $x + y = z$ and $x \cdot y = z$ are allowed as Σ_1-formulas.
(b) If A and B are Σ_1-formulas, then so are $A \wedge B$ and $A \vee B$.
(c) If A is a Σ_1-formula, then so is $\forall_{x<y} A$, which is an abbreviation for $\forall_x(\exists_z(x + Sz = y) \to A)$.
(d) If A is a Σ_1-formula, then so is $\exists_x A$.

COROLLARY. *Every recursive function is representable in Q by a Σ_1-formula in the language $\mathcal{L}_1$.*

PROOF. This can be seen immediately by inspecting the proof of the theorem above on weak arithmetical theories. Only notice that because of the equality axioms $\exists_z(x+Sz = y)$ is equivalent to $\exists_z\exists_w(Sz = w \wedge x+w = y)$ and $A(0)$ is equivalent to $\exists_x(0 = x \wedge A(x))$. □

3.4. Unprovability of consistency

We have seen in the theorem of Gödel-Rosser how, for every axiomatized consistent theory T safisfying certain weak assumptions, we can construct

an undecidable sentence A meaning "For every proof of me there is a shorter proof of my negation". Because A is unprovable, it is clearly true.

Gödel's second incompleteness theorem provides a particularly interesting alternative to A, namely a formula Con_T expressing the consistency of T. Again it turns out to be unprovable and therefore true. We shall prove this theorem in a sharpened form due to Löb.

3.4.1. Σ_1-completeness. We prove an auxiliary proposition, expressing the completeness of Q with respect to Σ_1-formulas.

LEMMA (Σ_1-completeness). *Let $A(x_1, \ldots, x_n)$ be a Σ_1-formula of the language $\mathcal{L}_1$. Assume that $\mathcal{N}_1 \models A(\underline{a_1}, \ldots, \underline{a_n})$ where $\mathcal{N}_1$ is the standard model of $\mathcal{L}_1$. Then $Q \vdash A(\underline{a_1}, \ldots, \underline{a_n})$.*

PROOF. By induction on the Σ_1-formulas of the language $\mathcal{L}_1$. For atomic formulas, the cases have been dealt with either in the earlier parts of the proof of the theorem above on weak arithmetical theories, or (for $x + y = z$ and $x \cdot y = z$) they follow from the recursion equations (28) - (31).

Cases $A \wedge B$, $A \vee B$. The claim follows immediately from the induction hypothesis.

Case $\forall_{x<y} A(x, y, z_1, \ldots, z_n)$; for simplicity assume $n = 1$. Suppose $\mathcal{N}_1 \models (\forall_{x<y} A)(\underline{b}, \underline{c})$. Then also $\mathcal{N}_1 \models A(\underline{i}, \underline{b}, \underline{c})$ for each $i < b$ and hence by induction hypothesis $Q \vdash A(\underline{i}, \underline{b}, \underline{c})$. Now by the theorem above on Robinson's Q

$$Q \vdash \forall_{x<\underline{b}}(x = \underline{0} \vee \cdots \vee x = \underline{b-1}),$$

hence

$$Q \vdash (\forall_{x<y} A)(\underline{b}, \underline{c}).$$

Case $\exists_x A(x, y_1, \ldots, y_n)$; for simplicity again take $n = 1$. Assume $\mathcal{N}_1 \models (\exists_x A)(\underline{b})$. Then $\mathcal{N}_1 \models A(\underline{a}, \underline{b})$ for some $a \in \mathbb{N}$, hence by induction hypothesis $Q \vdash A(\underline{a}, \underline{b})$ and therefore $Q \vdash (\exists_x A)(\underline{b})$. □

3.4.2. Derivability conditions. Let T be an axiomatized consistent theory with $T \supseteq Q$, and let $\mathrm{Prf}_T(p, z)$ be a Σ_1-formula of the language $\mathcal{L}_1$ which represents in Robinson's theory Q the recursive relation "a is the Gödel number of a proof in T of the formula with Gödel number b". Consider the following $\mathcal{L}_1$-formulas:

$$\begin{aligned} \mathrm{Thm}_T(x) &:= \exists_y \mathrm{Prf}_T(y, x), \\ \mathrm{Con}_T &:= \neg\exists_y \mathrm{Prf}_T(y, \underline{\ulcorner \bot \urcorner}). \end{aligned}$$

Then $\mathrm{Thm}_T(x)$ defines in $\mathcal{N}_1$ the set of formulas provable in T, and we have $\mathcal{N}_1 \models \mathrm{Con}_T$ if and only if T is consistent. We write $\Box A$ for $\mathrm{Thm}_T(\underline{\ulcorner A \urcorner})$;

hence Con_T can be written $\neg\Box\bot$. Now suppose, in addition, that T satisfies the following two *derivability conditions*, due to Hilbert and Bernays (1939):

$$T \vdash \Box A \to \Box\Box A, \tag{34}$$

$$T \vdash \Box(A \to B) \to \Box A \to \Box B. \tag{35}$$

(34) formalizes Σ_1-completeness of the theory T for closed formulas, and (35) is a formalization of its closure under modus ponens (i.e., $\to^-$). The derivability conditions place further restrictions on the theory T and its proof predicate Prf_T. We check them under the assumption that Prf_T is as defined earlier. (There are non-standard ways of coding proofs which lead to various "pathologies" - see, e.g., Feferman (1960)).

The formalized version of modus ponens is easy to see, assuming that T can be conservatively extended to include a "proof-term" $t(y, y')$ such that one may prove

$$\mathrm{Prf}_T(y, \underline{\ulcorner A \to B \urcorner}) \to \mathrm{Prf}_T(y', \underline{\ulcorner A \urcorner}) \to \mathrm{Prf}_T(t(y, y'), \underline{\ulcorner B \urcorner})$$

for then (35) follows immediately by quantifier rules.

(34) is harder. A detailed proof requires a great deal of syntactic machinery to do with the construction of proof terms, as above, acting on Gödel numbers so as to mimic the various rules inside T. We merely content ourselves here with a short indication of why (34) holds; this should be sufficient to convince the reader of its validity.

As we have seen at the beginning of this chapter, the elementary functions are provably recursive and so we may take their definitions as having been added conservatively. Working informally "inside" T one shows, by induction on y, that

$$\mathrm{Prf}_T(y, \underline{\ulcorner A \urcorner}) \to \mathrm{Prf}_T(f(y), \underline{\ulcorner \Box A \urcorner})$$

where f is elementary. Then (34) follows by the quantifier rules.

If y is the Gödel number of a derivation (in T) consisting of an axiom A then there will be a term t, elementarily computable from y, such that $\mathrm{Prf}_T(t, \underline{\ulcorner A \urcorner})$ and hence $\Box A$ are derivable in T. This derivation may be syntactically complex, but it will essentially consist of checking that t, as a Gödel number, encodes the right thing. Thus the derivation of $\Box A$ has a fixed Gödel number (depending on t and hence y) and this is what we take as the value of $f(y)$.

If y is the Gödel number of a derivation of A in which one of the rules is finally applied, say to premises A' and A'', then there will be $y', y'' < y$ such that $\mathrm{Prf}_T(y', \underline{\ulcorner A' \urcorner})$ and $\mathrm{Prf}_T(y'', \underline{\ulcorner A'' \urcorner})$. By the induction hypothesis, $f(y')$ and $f(y'')$ will be the Gödel numbers of T-derivations of $\Box A'$ and $\Box A''$, and as in the modus-ponens case above, there will be a fixed derivation which combines these two into a new derivation of $\Box A$. We take, as the value $f(y)$,

the Gödel number of this final derivation, computable from $f(y')$ and $f(y'')$ by applying some additional (sub-elementary) coding corresponding to the additional steps from $\Box A'$ and $\Box A''$ to $\Box A$.

The function f will be definable from elementary functions by a course-of-values recursion in which the recursion steps are in fact computed sub-elementarily. Therefore it will be a limited course-of-values recursion and, by a result in Chapter 2, f will therefore be elementary as required.

THEOREM (Gödel's second incompleteness theorem). *Let T be an axiomatized consistent extension of Q, satisfying the derivability conditions* (34) *und* (35). *Then $T \nvdash \mathrm{Con}_T$.*

PROOF. Let $C := \bot$ in Löb's theorem below, which is a generalization of Gödel's original result. □

THEOREM (Löb). *Let T be an axiomatized consistent extension of Q satisfying the derivability conditions* (34) *and* (35). *Then for any closed $\mathcal{L}_1$-formula C, if $T \vdash \Box C \to C$, then already $T \vdash C$.*

PROOF. Assume $T \vdash \Box C \to C$. We must show $T \vdash C$. Choose A by the fixed point lemma such that

$$Q \vdash A \leftrightarrow (\Box A \to C). \tag{36}$$

First we show $T \vdash \Box A \to C$. We obtain

$T \vdash A \to \Box A \to C$	by (36)
$T \vdash \Box(A \to \Box A \to C)$	by Σ_1-completeness
$T \vdash \Box A \to \Box(\Box A \to C)$	by (35)
$T \vdash \Box A \to \Box\Box A \to \Box C$	again by (35)
$T \vdash \Box A \to \Box C$	since $T \vdash \Box A \to \Box\Box A$ by (34).

Therefore the assumption $T \vdash \Box C \to C$ implies $T \vdash \Box A \to C$. Hence $T \vdash A$ by (36), and then $T \vdash \Box A$ by Σ_1-completeness. But $T \vdash \Box A \to C$ as we have just shown, therefore $T \vdash C$. □

REMARK. It follows that if T is any axiomatized consistent extension of Q satisfying the derivability conditions (34) und (35), then the reflection schema

$$\Box C \to C \quad \text{for closed } \mathcal{L}_1\text{-formulas } C$$

is not derivable in T. For by Löb's theorem, it cannot be derivable when C is underivable.

CHAPTER 4

Initial Cases of Transfinite Induction

The goal here is to study the in a sense "most complex" proofs in first-order arithmetic. The main tool for proving theorems in arithmetic is clearly the induction schema

$$A(0) \to \forall_x(A(x) \to A(Sx)) \to \forall_x A(x).$$

An equivalent form of this schema is "course-of-values" or cumulative induction

$$\forall_x(\forall_{y<x} A(y) \to A(x)) \to \forall_x A(x).$$

Both schemes refer to the standard ordering of $\mathbb{N}$. It is tempting to try to strengthen arithmetic by allowing more general induction schemes, e.g., w.r.t. the lexicographical ordering of $\mathbb{N} \times \mathbb{N}$. Even more generally, let $\prec$ be a well-ordering of $\mathbb{N}$ and use *transfinite induction*:

$$\forall_x(\forall_{y\prec x} A(y) \to A(x)) \to \forall_x A(x).$$

It can be understood as

> Suppose the property $A(x)$ is "progressive", i.e., from the validity of $A(y)$ for all $y \prec x$ we can conclude that $A(x)$ holds. Then $A(x)$ holds for all x.

For which well-orderings this schema is derivable in arithmetic? We will discuss a classic result of Gentzen (1943) which in a sense answers this question completely. However, to state the result we have to be more explicit about the well-orderings used.

4.1. Ordinals below ε_0

We need some knowledge and notations for ordinals. This can be done without relying on set theory: we introduce an initial segment of the ordinals (the ones $< \varepsilon_0$) in a formal, combinatorial way, i.e., via ordinal notations based on "Cantor normal form". From now on "ordinal" means "ordinal notation".

Definition. We define

(a) α is an ordinal,
(b) $\alpha < \beta$ for ordinals α, β.

simultaneously by induction, as follows.

(a) If $\alpha_m, \ldots, \alpha_0$ are ordinals, $m \geq -1$ and $\alpha_m \geq \cdots \geq \alpha_0$ (where $\alpha \geq \beta$ means $\alpha > \beta$ or $\alpha = \beta$), then

$$\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}$$

is an ordinal. The empty sum (denoted by 0) is allowed.

(b) If $\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}$ and $\omega^{\beta_n} + \cdots + \omega^{\beta_0}$ are ordinals, then

$$\omega^{\alpha_m} + \cdots + \omega^{\alpha_0} < \omega^{\beta_n} + \cdots + \omega^{\beta_0}$$

iff there is an $i \geq 0$ such that $\alpha_{m-i} < \beta_{n-i}$, $\alpha_{m-i+1} = \beta_{n-i+1}$, $\ldots$, $\alpha_m = \beta_n$, or else $m < n$ and $\alpha_m = \beta_n$, $\ldots$, $\alpha_0 = \beta_{n-m}$.

We shall use the notation:

$$\begin{aligned} 1 &:= \omega^0, \\ k &:= \omega^0 + \cdots + \omega^0 \quad \text{with } k \text{ copies of } \omega^0, \\ \omega^\alpha k &:= \omega^\alpha + \cdots + \omega^\alpha \quad \text{with } k \text{ copies of } \omega^\alpha. \end{aligned}$$

The *level* of an ordinal is defined by $\mathrm{lev}(0) := 0$, $\mathrm{lev}(\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}) := \mathrm{lev}(\alpha_m) + 1$. For ordinals of level $k+1$ we have $\omega_k \leq \alpha < \omega_{k+1}$, where $\omega_0 := 0$, $\omega_1 := \omega^1$, $\omega_{k+1} := \omega^{\omega_k}$. We also write ω for ω_1.

LEMMA 4.1.1. *$<$ is a linear order with 0 the least element.*

PROOF. By induction on the levels. □

EXAMPLE.

$$\begin{aligned} &0 < 1 < 2 \cdots < \omega < \omega + 1 \cdots < \omega 2 < \omega 2 + 1 \cdots < \omega 3 \cdots < \omega^2 \\ &< \omega^2 + 1 \cdots < \omega^2 + \omega \cdots < \omega^3 \cdots < \omega^\omega = \omega_2 \cdots < \omega_3 \cdots \end{aligned}$$

DEFINITION (Addition of ordinals).

$$(\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}) + (\omega^{\beta_n} + \cdots + \omega^{\beta_0}) := \omega^{\alpha_m} + \cdots + \omega^{\alpha_i} + \omega^{\beta_n} + \cdots + \omega^{\beta_0}$$

where i is minimal such that $\alpha_i \geq \beta_n$.

LEMMA 4.1.2. *$+$ is an associative operation which is strictly monotonic in the second argument and weakly monotonic in the first argument.*

PROOF. Exercise. □

REMARK. $+$ is *not* commutative:

$$1 + \omega = \omega \neq \omega + 1.$$

There is also a commutative version of addition, the *natural sum* (or Hessenberg sum). It is defined by

$$(\omega^{\alpha_m} + \cdots + \omega^{\alpha_0}) \# (\omega^{\beta_n} + \cdots + \omega^{\beta_0}) := \omega^{\gamma_{m+n+1}} + \cdots + \omega^{\gamma_0},$$

where $\gamma_{m+n+1}, \ldots, \gamma_0$ is a decreasing permutation of $\alpha_m, \ldots, \alpha_0, \beta_n, \ldots, \beta_0$. It is easy to see that $\#$ is associative, commutative and strictly monotonic in both arguments.

How ordinals of the form $\beta + \omega^\alpha$ can be approximated from below? First note that

$$\delta < \alpha \to \beta + \omega^\delta k < \beta + \omega^\alpha.$$

For any $\gamma < \beta + \omega^\alpha$ we can find a $\delta < \alpha$ and a k such that

$$\gamma < \beta + \omega^\delta k.$$

It is easy to code ordinals $< \varepsilon_0$ bijectively by natural numbers:

$$\mathrm{o}(\ulcorner \alpha \urcorner) = \alpha \quad \text{and} \quad \ulcorner \mathrm{o}(x) \urcorner = x$$

such that relations and operations on ordinals transfer to elementary relations and operations on $\mathbb{N}$. Abbreviations:

$$\begin{aligned} x \prec y &:= \mathrm{o}(x) < \mathrm{o}(y), \\ \omega^x &:= \ulcorner \omega^{\mathrm{o}(x)} \urcorner, \\ x \oplus y &:= \ulcorner \mathrm{o}(x) + \mathrm{o}(y) \urcorner, \\ xk &:= \ulcorner \mathrm{o}(x) k \urcorner, \\ \omega_k &:= \ulcorner \omega_k \urcorner. \end{aligned}$$

4.2. Provability of initial cases of transfinite induction

We will derive initial cases of transfinite induction in arithmetic:

$$\forall_x (\forall_{y \prec x} Py \to Px) \to \forall_{x \prec a} Px$$

for some number a and a predicate symbol P, where $\prec$ is the standard order of order type ε_0 defined before.

Remark. Gentzen (1943) proved that this result is optimal in the sense that for the full system of ordinals $< \varepsilon_0$ the principle

$$\forall_x (\forall_{y \prec x} Py \to Px) \to \forall_x Px$$

of transfinite induction is underivable. However, we will not present a proof in these notes.

By an *arithmetical system* $\mathbf{Z}$ we mean a theory based on minimal logic in the $\forall\to$-language (including equality axioms) such that

(a) The language of **Z** consists of a fixed supply of function and relation constants assumed to denote computable functions and relations on the non-negative integers.
(b) Among the function constants there is a constant S for the successor function and 0 for (the 0-place function) zero.
(c) Among the relation constants we have $=$, P and also $\prec$ for the ordering of type ε_0 of $\mathbb{N}$, as introduced before.
(d) *Terms* are built up from object variables x, y, z by $f(t_1, \ldots, t_m)$, where f is a function constant.
(e) We identify closed terms which have the same value; this expresses that each function constant is computable.
(f) Terms of the form $S(S(\ldots S0\ldots))$ are called *numerals*. Notation: S^n0 or $\underline{n}$ or just n.
(g) *Formulas* are built up from atomic formulas $R(t_1, \ldots, t_m)$, with R a relation constant, by $A \to B$ and $\forall_x A$.

The axioms of **Z** are

- Compatibility of equality

$$x = y \to A(x) \to A(y),$$

- the *Peano axioms*, i.e., the universal closures of

$$Sx = Sy \to x = y, \tag{37}$$

$$Sx = 0 \to A, \tag{38}$$

$$A(0) \to \forall_x(A(x) \to A(Sx)) \to \forall_x A(x), \tag{39}$$

with $A(x)$ an arbitrary formula.
- $R\vec{n}$ whenever $R\vec{n}$ is true (to express that R is computable).
- Irreflexivity and transitivity for $\prec$

$$x \prec x \to A,$$
$$x \prec y \to y \prec z \to x \prec z$$

Further axioms – following Schütte – are the universal closures of

$$x \prec 0 \to A, \tag{40}$$

$$z \prec y \oplus \omega^0 \to (z \prec y \to A) \to (z = y \to A) \to A, \tag{41}$$

$$x \oplus 0 = x, \tag{42}$$

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z, \tag{43}$$

$$0 \oplus x = x, \tag{44}$$

$$\omega^x 0 = 0, \tag{45}$$

$$\omega^x(Sy) = \omega^x y \oplus \omega^x, \tag{46}$$

$$z \prec y \oplus \omega^{Sx} \to z \prec y \oplus \omega^{e(x,y,z)} m(x,y,z), \tag{47}$$

$$z \prec y \oplus \omega^{Sx} \to e(x,y,z) \prec Sx, \tag{48}$$

$$y \oplus \omega^{x} \prec y \oplus \omega^{Sx}, \tag{49}$$

where $\oplus$, $\lambda_{x,y}(\omega^x y)$, e and m denote function constants and A is any formula. These axioms are formal counterparts to the properties of the ordinal notations observed above.

THEOREM 4.2.1 (Provable initial cases of transfinite induction in **Z**). *Transfinite induction up to ω_n, i.e., for arbitrary $A(x)$ the formula*

$$\forall_x(\forall_{y \prec x} A(y) \to A(x)) \to \forall_{x \prec \omega_n} A(x),$$

is derivable in **Z**.

PROOF. To every formula $A(x)$ we assign a formula $A^+(x)$ (with respect to a fixed variable x) by

$$A^+(x) := \forall_y(\forall_{z \prec y} A(z) \to \forall_{z \prec y \oplus \omega^x} A(z)).$$

We first show

$$\text{If } A(x) \text{ is progressive, then } A^+(x) \text{ is progressive,}$$

where "$B(x)$ is *progressive*" means $\forall_x(\forall_{y \prec x} B(y) \to B(x))$. Assume that $A(x)$ is progressive and

$$\forall_{y \prec x} A^+(y). \tag{50}$$

Our goal is $A^+(x) := \forall_y(\forall_{z \prec y} A(z) \to \forall_{z \prec y \oplus \omega^x} A(z))$. Assume

$$\forall_{z \prec y} A(z), \tag{51}$$

$$z \prec y \oplus \omega^x. \tag{52}$$

We have to show $A(z)$.

Case $x = 0$. Then $z \prec y \oplus \omega^0$. By (41):

$$z \prec y \oplus \omega^0 \to (z \prec y \to A) \to (z = y \to A) \to A$$

it suffices to derive $A(z)$ from $z \prec y$ as well as from $z = y$. If $z \prec y$, then $A(z)$ follows from (51), and if $z = y$, then $A(z)$ follows from (51) and the progressivness of $A(x)$.

Case Sx. From $z \prec y \oplus \omega^{Sx}$ we obtain $z \prec y \oplus \omega^{e(x,y,z)} m(x,y,z)$ by (47) and $e(x,y,z) \prec Sx$ by (48). By (50) we have $A^+(e(x,y,z))$, i.e.

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} v} A(u) \to \forall_{u \prec (y \oplus \omega^{e(x,y,z)} v) \oplus \omega^{e(x,y,z)}} A(u)$$

and hence, using (43) and (46)

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} v} A(u) \to \forall_{u \prec y \oplus \omega^{e(x,y,z)} (Sv)} A(u).$$

Also from (51) and (45), (42) we obtain

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)}0} A(u).$$

By induction:

$$\forall_{u \prec y \oplus \omega^{e(x,y,z)} m(x,y,z)} A(u)$$

and hence $A(z)$, since $z \prec y \oplus \omega^{e(x,y,z)} m(x,y,z)$. This is a consequence of (47), whose premise $z \prec y \oplus \omega^{Sx}$ follows via (49) from (52).

Next we show, by induction on n, how to derive

$$\forall_x(\forall_{y \prec x} A(y) \to A(x)) \to \forall_{x \prec \omega_n} A(x) \quad \text{for arbitrary } A(x).$$

Assume the left hand side, i.e., that $A(x)$ is progressive.

Case 0. Then $x \prec \omega^0$ and hence $x \prec 0 \oplus \omega^0$ by (44). By (41) it suffices to derive $A(x)$ from $x \prec 0$ as well as from $x = 0$. Now $x \prec 0 \to A(x)$ holds by (40), and $A(0)$ then follows from the progressivness of $A(x)$.

Case $n{+}1$. Since $A(x)$ is progressive, also $A^+(x)$ is. By IH: $\forall_{x \prec \omega_n} A^+(x)$, hence $A^+(\omega_n)$ since $A^+(x)$ is progressive. By definition of $A^+(x)$ (with (40): $x \prec 0 \to A$ and (44): $0 \oplus x = x$) we obtain $\forall_{z \prec \omega^{\omega_n}} A(z)$. □

Remark. In the induction step we derived transfinite induction up to ω_{n+1} for $A(x)$ from transfinite induction up to ω_n for $A^+(x)$. Define the *level* of a formula by

$$\begin{aligned} \mathrm{lev}(R\vec{t}\,) &:= 0, \\ \mathrm{lev}(A \to B) &:= \max(\mathrm{lev}(A) + 1, \mathrm{lev}(B)), \\ \mathrm{lev}(\forall_x A) &:= \max(1, \mathrm{lev}(A)). \end{aligned}$$

Then $\mathrm{lev}(A^+(x)) = \mathrm{lev}(A(x)) + 1$. Hence to prove transfinite induction up to ω_n, the induction scheme in **Z** is used for formulas of level n.

4.3. Iteration operators of higher types

We have just seen that the strength of the induction scheme increases with the level of the formula proved by induction. A similar phenomenon occurs when one considers types instead of formulas, and iteration (a special case of recursion) instead of induction. Such operators have a similar relation to ordinals $<\varepsilon_0$.

Definition. An ordinal $\omega^{\alpha_n} + \cdots + \omega^{\alpha_0}$ is a *successor* if $\alpha_0 = 0$. It is a *limit* if α_0 it is neither 0 nor a successor. For every limit $\lambda = \omega^{\alpha_n} + \cdots + \omega^{\alpha_0}$ we define its *fundamental sequence* $\lambda[x]$ by

$$\lambda[x] := \begin{cases} \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0 - 1} \cdot x & \text{if } \alpha_0 \text{ is a successor} \\ \omega^{\alpha_n} + \ldots + \omega^{\alpha_1} + \omega^{\alpha_0[x]} & \text{if } \alpha_0 \text{ is a limit.} \end{cases}$$

EXAMPLES.

$$\begin{aligned}\omega[x] &= x,\\ (\omega+\omega)[x] &= \omega + x,\\ \omega^2[x] &= \omega x,\\ \omega^3[x] &= \omega^2 x,\\ \omega^\omega[x] &= \omega^x.\end{aligned}$$

DEFINITION (Extended Grzegorczyk hierarchy $(F_\alpha)_{\alpha<\varepsilon_0}$).

$$\begin{aligned}F_0(x) &:= 2^x,\\ F_{\alpha+1}(x) &:= F_\alpha^{(x)}(x) \quad (F_\alpha^{(x)}\ x\text{-th iterate of } F_\alpha),\\ F_\lambda(x) &:= F_{\lambda[x]}(x).\end{aligned}$$

We also define $F_{\varepsilon_0}(x) := F_{\omega_x}(x)$.

REMARK. F_ω is a variant of the Ackermann function (1940), and the F_n for $n < \omega$ were (essentially) defined and studied by Grzegorczyk (1953).

LEMMA 4.3.1. *The function F_1 is not an elementary function, but its graph is an elementary relation.*

PROOF. That F_1 is not elementary was essentially proved as a lemma in Section 2.2.1. The see that the graph of F_1 is elementary observe that

$$F_1(x) = y \leftrightarrow \exists_z\big((z)_0 = 0 \wedge \forall_{i<x}((z)_{i+1} = 2^{(z)_i}) \wedge (z)_x = y\big).$$

Now it suffices to prove that z can be bounded by an elementary function in x and y. But since F_0 is increasing we can bound z by $\langle y,\dots,y\rangle$ with x occurrences of y, and by a lemma in Section 2.2.5 we have

$$\langle \underbrace{y,\dots,y}_{x}\rangle < (y+1)^{2^x}. \qquad \square$$

Using similar arguments one can prove that all functions F_α for $\alpha < \varepsilon_0$ have elementary graphs.

Let T be a theory in a language containing $0, S$ with the property that every elementary relation is representable in T. We call a function $f\colon \mathbb{N} \to \mathbb{N}$ *provably recursive* in T if we have a formula A_f representing the graph of f such that

$$T \vdash \forall_x \exists_y A_f(x,y).$$

In standard arithmetical systems like **Z** one can prove that all functions F_α for $\alpha < \varepsilon_0$ are provably recursive, with methods similar to what we used in Section 4.2. Again ε_0 is a sharp bound: F_{ε_0} is not be provably recursive.

We can characterize $(F_\alpha)_{\alpha<\varepsilon_0}$ by higher type iteration. To this end we extend the definition of the functions F_α into higher types.

Types are generated from the base type $\mathbb{N}$ by the formation of *function types* $\tau \to \sigma$. The *level* of a type is defined similar to the level of a formula (in Section 4.2), by

$$\begin{aligned} \mathrm{lev}(\mathbb{N}) &:= 0, \\ \mathrm{lev}(\tau \to \sigma) &:= \max(\mathrm{lev}(\tau) + 1, \mathrm{lev}(\sigma)). \end{aligned}$$

It is convenient here to introduce *integer types* ρ_n:

$$\begin{aligned} \rho_0 &:= \mathbb{N}, \\ \rho_{n+1} &:= \rho_n \to \rho_n. \end{aligned}$$

If $x_0, \dots, x_{n+1}$ are of integer types $\rho_0, \dots, \rho_{n+1}$, then we can form $x_{n+1}(x_n)$ (of type ρ_n) and so on, finally $x_{n+1}(x_n) \dots (x_0)$, or shortly $x_{n+1}(x_n, \dots, x_0)$. Note that $\mathrm{lev}(\rho_n) = n$.

We define F^{n+1}_α of type ρ_{n+1} for $\alpha < \varepsilon_0$:

$$\begin{aligned} F^{n+1}_0(x_n, \dots, x_0) &:= \begin{cases} 2^{x_0} & \text{if } n = 0 \\ x_n^{(x_0)}(x_{n-1}, \dots, x_0) & \text{otherwise.} \end{cases} \\ F^{n+1}_{\alpha+1}(x_n, \dots, x_0) &:= (F^{n+1}_\alpha)^{(x_0)}(x_n, \dots, x_0), \\ F^{n+1}_\lambda(x_n, \dots, x_0) &:= F^{n+1}_{\lambda[x_0]}(x_n, \dots, x_0). \end{aligned}$$

Here $x_n^{(y)}(x_{n-1}, \dots, x_0)$ denotes $I(y, x_n, \dots, x_0)$ with an *iteration* functional I of type $\mathbb{N} \to \rho_n \to \rho_{n-1} \to \dots \to \rho_0 \to \rho_0$ defined by

$$\begin{aligned} I(0, y, z) &:= z, \\ I(x+1, y, z) &:= y(I(x, y, z)). \end{aligned}$$

THEOREM 4.3.2. *For $n \geq 1$ we have*

$$F^{n+1}_\alpha(F^n_\beta) = F^n_{\beta + \omega^\alpha}$$

provided $\beta + \omega^\alpha = \beta \# \omega^\alpha$, i.e., in the Cantor normal form of β the last summand ω^{β_0} (if it exists) has an exponent $\beta_0 \geq \alpha$.

PROOF. By induction on α. *Case* $\alpha = 0$.

$$\begin{aligned} F^{n+1}_0(F^n_\beta, x_{n-1}, \dots, x_0) &= (F^n_\beta)^{(x_0)}(x_{n-1}, \dots, x_0) \\ &= F^n_{\beta+1}(x_{n-1}, \dots, x_0). \end{aligned}$$

Case α successor.

$$\begin{aligned} F^{n+1}_\alpha(F^n_\beta, x_{n-1}, \dots, x_0) &= (F^{n+1}_{\alpha-1})^{(x_0)}(F^n_\beta, x_{n-1}, \dots, x_0) \\ &= F^n_{\beta + \omega^{\alpha-1} \cdot x_0}(x_{n-1}, \dots, x_0) \quad \text{by IH} \\ &:= F^n_{(\beta+\omega^\alpha)[x_0]}(x_{n-1}, \dots, x_0) \\ &:= F^n_{\beta+\omega^\alpha}(x_{n-1}, \dots, x_0). \end{aligned}$$

Case α limit.

$$\begin{aligned} F^{n+1}_{\alpha}(F^n_\beta, x_{n-1}, \ldots, x_0) &= F^{n+1}_{\alpha[x_0]}(F^n_\beta, x_{n-1}, \ldots, x_0) \\ &= F^n_{\beta+\omega^{\alpha[x_0]}}(x_{n-1}, \ldots, x_0) \quad \text{by IH} \\ &= F^n_{(\beta+\omega^\alpha)[x_0]}(x_{n-1}, \ldots, x_0) \\ &= F^n_{\beta+\omega^\alpha}(x_{n-1}, \ldots, x_0). \end{aligned}$$ □

The result just proved indicates the computational complexity involved in the use of finite types. The functionals $(F^{n+1}_\alpha)_{\alpha<\varepsilon_0}$ and in particular the functions $(F^1_\alpha)_{\alpha<\varepsilon_0}$ can be built from *iteration* functionals (and $F_0(x) = 2^x$) by application alone. In the resulting representation of the functions $(F_\alpha)_{\alpha<\varepsilon_0}$ we do not need the fundamental sequences $\lambda[x]$. The application pattern for F_α corresponds to the Cantor normal form of α.

CHAPTER 5

Computational content of proofs

Mathematics differs from all other sciences by the fact that it provides proofs for its claims. In this chapter we study what we can do with proofs apart from assuring us of the truth of what they state.

Present technology makes it feasible to generate even large formal proofs with "proof assistants". These proof objects can be checked for logical correctness, independently of the used proof assistant: one only needs to check that the logical rules have been applied correctly.

Our underlying "minimal" logic is "constructive" in the sense that a "computationally relevant" (c.r.) predicate (for instance an existential statement $\exists_x A$ or a disjunction $A \vee B$) can only be proved by providing an example. Then from a proof of a c.r. statement we can extract a term which can be seen as a program representing the computational content of the proof. One can develop the theoretical concepts used to formulate and prove this "Soundness theorem".

Usage of a proof assistant makes it possible to automate this extraction process, and also to automatically generate a formal proof that the extracted term "realizes" the original statement.

5.1. Issues

We very briefly describe some of the issues involved when studying the computational content of proofs. First of all, since the subject is in its infancy it seems best to start with concrete rather than abstract mathematics.

5.1.1. Types. Mathematical objects have "types". Examples are "base types" $\mathbb{N}$, $\mathbb{Q}$ of natural or rational numbers, and the "function type" $\mathbb{N} \to \mathbb{N}$ of unary functions f on $\mathbb{N}$. The iteration operator I with $I(f, n, m) := f^{(n)}(m)$ has type $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N} \to \mathbb{N}$.

5.1.2. Approximations. Infinite objects are viewed as given by their (finite) approximations. Examples are real numbers, or functions $f \colon \mathbb{N} \to \mathbb{N}$.

5.1.3. Model. Objects of type τ are the "partial continuous functionals" of Scott (1982) and Ershov (1977). Continuity of a function f means

that for every approximation V of the value $f(x)$ there is an approximation U of the argument x such that $f[U]$ has more information than V.

5.1.4. Constants and predicates. We will need constants for functionals of possibly higher types, given by "computation rules". Examples: Double: $\mathbb{N} \to \mathbb{N}$ is defined by

$$\mathrm{Double}(0) = 0, \quad \mathrm{Double}(S(n)) = S(S(\mathrm{Double}(n))),$$

and iteration I: $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ by

$$I(f, 0, m) = m, \quad I(f, S(n), m) = f(I(f, n, m)).$$

Predicates of arity $\vec{\tau}$ (a list of types) are either predicate variables or predicate constants. A predicate variable is a place holder for a formula with some object variables bound (for example $\{\, n, m \mid n \le m \,\}$). A predicate constant is inductively or coinductively defined. This means that it is the least or greatest fixed point of some "clauses". An example of an inductive predicate is Even, the least fixed point of the clauses

$$\mathrm{Even}(0), \quad \forall_n(\mathrm{Even}(n) \to \mathrm{Even}(S(S(n)))).$$

An example of a coinductive predicate is ${}^{\mathrm{co}}\mathrm{Even}$, the greatest fixed point of the single clause

$$\forall_n({}^{\mathrm{co}}\mathrm{Even}(n) \to \exists_m(n = S(S(m)) \land {}^{\mathrm{co}}\mathrm{Even}(m))).$$

The logical connectives $\exists$, $\lor$ and $\land$ are viewed as inductively defined predicates with parameters.

5.1.5. Proof trees. To generate proofs we think of them as trees with "holes" to be filled. Each hole carries a formula (called "goal") and a "context" consisting of object and assumption variables. The task then is to "refine" the proof by filling a hole with a logical rule, which may bind one of the object or assumption variables. This process is well supported by an interactive proof assistant. Guidance by a previous proof idea is of course essential here.

5.1.6. Formulas as problems. In constructive mathematics proofs may have "computational content". According to Kolmogorov (1932) a formula can be seen as a problem, asking for a solution. A solution for a problem posed by an implication $A \to B$ is a computable functional mapping solutions of A into solutions of B.

5.1.7. Decoration. Sometimes the solution of a problem does not need all available input. We therefore mark (or "decorate") the sources of such computationally superfluous input – that is, some predicates – as "non-computational" (n.c.), and the other ones as "computationally relevant" (c.r.). More precisely, we mark predicates either as P^{c} or as P^{nc}.

5.1.8. Non-computational parts of proofs. Proofs use the rules of minimal logic for $\to$ and $\forall$, and some axioms. However, because of the presence of decorations we have an extra degree of freedom. By an *n.c. part* of a proof we mean a subproof with an n.c. end formula. Such n.c. parts will not contribute to the computational content of the whole proof, and hence we can ignore all decorations in those parts (i.e., use a modified notion of equality of formulas there).

5.1.9. Weak and non-computational existence and disjunction. We now have two forms of existential formulas without computational content: $\exists^{\mathrm{nc}}_x A$ and $\tilde{\exists}_x A$, where the latter abbreviates $\neg\forall_x\neg A$. It is easy to prove $\exists^{\mathrm{nc}}_x A \to \tilde{\exists}_x A$. However, for the other direction we need stability of $\exists^{\mathrm{nc}}_x A$:

$$\neg\neg\exists^{\mathrm{nc}}_x A \to \exists^{\mathrm{nc}}_x A. \tag{53}$$

Lemma 5.1.1 (WeakExToExNc). $\tilde{\exists}_x A \to \exists^{\mathrm{nc}}_x A$.

Proof. Because of (53) it suffices to prove $\tilde{\exists}_x A \to \neg\neg\exists^{\mathrm{nc}}_x A$. Since $\tilde{\exists}_x A$ abbreviates $\neg\forall_x\neg A$ is also suffices to prove $\neg\exists^{\mathrm{nc}}_x A \to \forall_x\neg A$. Assume the premise. Let x be arbitrary and assume A. The goal is falsity $\mathbf{F}$. Because of the premise we only need to prove $\exists^{\mathrm{nc}}_x A$. But this follows from A by the introduction rule for $\exists^{\mathrm{nc}}$. □

It seems reasonable to accept (53) as an axiom. Since it is n.c. this axiom does not affect the computational content of proofs using it. Its benefit is that we then can freely exchange $\exists^{\mathrm{nc}}$ and $\tilde{\exists}$, and use the one whose rules fit best for the problem at hand.

The two disjunctions $A \vee^{\mathrm{nc}} B$ and $A \mathbin{\tilde{\vee}} B$ are treated similarly, using stability of $A \vee^{\mathrm{nc}} B$ as an axiom:

$$\neg\neg(A \vee^{\mathrm{nc}} B) \to A \vee^{\mathrm{nc}} B. \tag{54}$$

5.2. An application in constructive analysis

We are interested in *exact real numbers*, as opposed to floating point numbers. The final goal is to develop the basics of real analysis in such a way that from a proof of an existence formula one can extract a program. For instance, from a proof of the intermediate value theorem we want to extract a program that, given an arbitrary error bound $\frac{1}{2^p}$, computes a rational x where the given function is zero up to the error bound.

5.2.1. Real numbers, continuous functions. The first thing to do is to build up the number systems: natural numbers in unary and binary notation, integers, rationals and reals. The latter are viewed as given by a Cauchy sequence of rationals together with a modulus. Equality of reals has to be defined accordingly.

We consider (uniformly) continuous functions on closed real intervals, with a modulus. Such functions are already determined by their values on rationals, and hence by data of type level 1. As a consequence, application $f(x)$ of a continuous function f to a real x needs to be defined separately. This can be done, in such a way that application is compatible with equality on reals, i.e.,

$$x =_{\mathbb{R}} y \to f(x) =_{\mathbb{R}} f(y).$$

Composition of continuous functions can be defined, and it has the expected properties.

5.2.2. Intermediate value theorem. The usual formulation is

> Let $f\colon I \to \mathbb{R}$ be continuous and $a < b$ rational numbers in I such that $f(a) \le 0 \le f(b)$. Then for every p we can find c with $a < c < b$ such that $|f(c)| \le \frac{1}{2^p}$.

A problem with the standard constructive proof is that the algorithm it provides is rather bad: in each case one has to partition the interval into as many pieces as the modulus of the continuous function requires for the given error bound, and then for each of these (many) pieces perform certain operations. This problem seems to be unavoidable, since our continuous function may be rather flat. However, we can do somewhat better if we assume a uniform *modulus of increase* (or lower bound on the slope) of f, that is, some $q \in \mathbb{Z}^+$ such that for all $c, d \in \mathbb{Q}$ and all $p \in \mathbb{Z}^+$

$$\frac{1}{2^p} \le d - c \to \frac{1}{2^{p+q}} \le f(d) - f(c).$$

Constructively, we cannot compare two reals, but we can compare every real with a nontrivial interval.

LEMMA 5.2.1 (ApproxSplit). *Let x, y, z be given and assume $x < y$. Then either $z \le y$ or $x \le z$.*

PROOF. Let $x := ((a_n)_n, M)$, $y := ((b_n)_n, N)$, $z := ((c_n)_n, L)$. Assume $x <_p y$, that is (by definition) $\frac{1}{2^p} \le b_n - a_n$ for $n := \max(M(p+2), N(p+2))$. Let $m := \max(n, L(p+2))$.

Case $c_m \le \frac{a_n + b_n}{2}$. We show $z \le y$. It suffices to prove $c_l \le b_l$ for $l \ge m$. This follows from

$$c_l \le c_m + \frac{1}{2^{p+2}} \le \frac{a_n + b_n}{2} + \frac{b_n - a_n}{4} = b_n - \frac{b_n - a_n}{4} \le b_n - \frac{1}{2^{p+2}} \le b_l.$$

Case $c_m \not\le \frac{a_n + b_n}{2}$. We show $x \le z$. This follows from $a_l \le c_l$ for $l \ge m$:

$$a_l \le a_n + \frac{1}{2^{p+2}} \le a_n + \frac{b_n - a_n}{4} = \frac{a_n + b_n}{2} - \frac{b_n - a_n}{4} \le c_m - \frac{1}{2^{p+2}} \le c_l. \quad \square$$

Notice that the boolean object determining whether $z \leq y$ or $x \leq z$ depends on the representation of x, y and z. In particular this assignment is *not* compatible with our equality relation.

We begin with an auxiliary lemma, which from a "correct" interval $c < d$ (that is, $f(c) \leq 0 \leq f(d)$ and $\frac{1}{2^p} \leq d - c$) constructs a new one $c_1 < d_1$ with $d_1 - c_1 = \frac{1}{2}(d - c)$.

LEMMA 5.2.2 (IVTAux). *Let $f\colon I \to \mathbb{R}$ be continuous, with a uniform modulus q of increase. Let $a < b$ be rational numbers in I such that $a \leq c < d \leq b$, say $\frac{1}{2^p} < d - c$, and $f(c) \leq 0 \leq f(d)$. Then we can construct c_1, d_1 with $d_1 - c_1 = \frac{1}{2}(d - c)$, such that again $a \leq c \leq c_1 < d_1 \leq d \leq b$ and $f(c_1) \leq 0 \leq f(d_1)$.*

PROOF. Let $b_0 = c$ and $b_{n+1} = b_n + \frac{1}{4}(d - c)$ for $n \leq 3$, hence $b_4 = d$. From $\frac{1}{2^p} < d - c$ we obtain $\frac{1}{2^{p+2}} \leq b_{n+1} - b_n$, so $f(b_n) <_{p+2+q} f(b_{n+1})$.

First compare 0 with the proper interval $f(b_1) < f(b_2)$, using ApproxSplit. In case $0 \leq f(b_2)$ let $c_1 = b_0 = c$ and $d_1 = b_2$. In case $f(b_1) \leq 0$ compare 0 with the proper interval $f(b_2) < f(b_3)$, using ApproxSplit again. In case $0 \leq f(b_3)$ let $c_1 = b_1$ and $d_1 = b_3$. In case $f(b_2) \leq 0$ let $c_1 = b_2$ and $d_1 = b_4 = d$. □

THEOREM 5.2.3 (IVT). *Let $f\colon I \to \mathbb{R}$ be continuous, with a uniform modulus of increase. Let $a < b$ be rational numbers in I such that $f(a) \leq 0 \leq f(b)$. Then we can find $x \in [a, b]$ such that $f(x) = 0$.*

PROOF. Iterating the construction in Lemma 5.2.2 (IVTAux), we construct two sequences $(c_n)_n$ and $(d_n)_n$ of rationals such that for all n

$$\begin{aligned} &a = c_0 \leq c_1 \leq \cdots \leq c_n < d_n \leq \cdots \leq d_1 \leq d_0 = b, \\ &f(c_n) \leq 0 \leq f(d_n), \\ &d_n - c_n = \frac{1}{2^n}(b - a). \end{aligned}$$

Let x, y be given by the Cauchy sequences $(c_n)_n$ and $(d_n)_n$ with the obvious modulus. As f is continuous, $f(x) = 0 = f(y)$ for the real number $x = y$. □

As an example consider the function $x^2 - 2$ on the real interval $[1, 2]$. Applying the theorem to this continuous function gives us a constructive proof the the existence of the square root of 2. This proof is implemented in Minlog, and a program (i.e., a term in the term language of the underlying theory) representing its computational content has been extracted. Evaluating this term up to a fixed precision gives us an approximation of the square root of 2 of the required accuracy.

Bibliography

W. Ackermann. Zur Widerspruchsfreiheit der Zahlentheorie. *Math. Annalen*, 117:162–194, 1940.

E. W. Beth. Semantic construction of intuitionistic logic. *Medelingen de KNAW N.S.*, 19(11), 1956.

E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer Verlag, Berlin, Heidelberg, New York, 1997.

H. B. Curry. Grundlagen der kombinatorischen Logik. *Amer. J. Math.*, 52: 509–536, 789–834, 1930.

N. G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. *Indagationes Math.*, 34:381–392, 1972.

Y. L. Ershov. Model C of partial continuous functionals. In R. Gandy and M. Hyland, editors, *Logic Colloquium 1976*, pages 455–467. North-Holland, Amsterdam, 1977.

S. Feferman. Arithmetization of metamathematics in a general setting. *Fundamenta Mathematicae*, XLIX:35–92, 1960.

G. Gentzen. Untersuchungen über das logische Schließen I, II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.

G. Gentzen. Beweisbarkeit und Unbeweisbarkeit von Anfangsfällen der transfiniten Induktion in der reinen Zahlentheorie. *Mathematische Annalen*, 119:140–161, 1943.

A. Grzegorczyk. *Some Classes of Recursive Functions.* Rozprawy Matematyczne, Warszawa, 1953.

A. Heyting, editor. *Constructivity in Mathematics*, 1959. North-Holland, Amsterdam.

D. Hilbert and P. Bernays. *Grundlagen der Mathematik*, volume II. Springer Verlag, Berlin, Heidelberg, New York, 1939.

W. A. Howard. The formulae–as–types notion of construction. In J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

I. Johansson. Der Minimalkalkül, ein reduzierter intuitionistischer Formalismus. *Compositio Mathematica*, 4:119–136, 1937.

L. Kalmár. Ein einfaches Beispiel für ein unentscheidbares Problem (Hungarian, with German summary). *Mat. Fiz. Lapok*, 50:1–23, 1943.

A. N. Kolmogorov. Zur Deutung der intuitionistischen Logik. *Math. Zeitschr.*, 35:58–65, 1932.

P. Martin-Löf. Infinite terms and a system of natural deduction. *Compositio Mathematica*, 24(1):93–103, 1972.

R. M. Robinson. An essentially undecidable axiom system. In *Proceedings of the International Congress of Mathematicians (Cambridge 1950)*, volume I, pages 729–730, 1950.

H. Schwichtenberg and S. S. Wainer. *Proofs and Computations*. Perspectives in Logic. Association for Symbolic Logic and Cambridge University Press, 2012.

D. Scott. Domains for denotational semantics. In E. Nielsen and E. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *LNCS*, pages 577–613. Springer Verlag, Berlin, Heidelberg, New York, 1982.

J. C. Shepherdson and H. E. Sturgis. Computability of recursive functions. *J. Ass. Computing Machinery*, 10:217–255, 1963.

A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*. Cambridge University Press, second edition, 2000.

H. Weyl. Über die neue Grundlagenkrise der Mathematik. *Mathematische Zeitschrift*, 10, 1921.

Index