An arithmetic for polynomial-time computation

Helmut Schwichtenberg*

July 25, 2005

Abstract

We define a restriction **LHA** of Heyting arithmetic **HA** with the property that all extracted programs are feasible. The restrictions consist in *linearity* and *ramification* requirements.

1 Introduction

It is well known that it is undecidable in general whether a given program meets its specification. In contrast, it can be checked easily by a machine whether a formal proof is correct, and from a constructive proof one can automatically extract a corresponding program, which by its very construction is correct as well. This – at least in principle – opens a way to produce correct software, e.g. for safety-critical applications. Moreover, programs obtained from proofs are "commented" in a rather extreme sense. Therefore it is easy to maintain them, and also to adapt them to particular situations.

Clearly efficiency of the extracted program is a major concern for such a project. The goal of the present paper is to present a constructive arithmetic which ensures that all extracted programs are polynomial-time computable, and at the same time is flexible enough to allow for the representation of particular polynomial-time algorithms, not just polynomial-time functions.

Recursion in all finite types was introduced by Hilbert [14], the system later becoming known as Gödel's system **T** [13]. The value computed by a higher type recursion can be any functional, which is to say a mapping that takes other mappings as arguments and produces a new mapping. Correspondingly one defines a type system of functions and functionals over some ground types. This recursion in higher types has long been viewed as a powerful scheme unsuitable for describing small complexity classes such as polynomial time. It is well known that ramification can be used to restrict higher type recursion. However, to characterize the very small class

*Mathematisches Institut der Universität München, Theresienstraße 39, D-80333 München, Germany. Phone +49 89 2180 4413, Fax +49 89 280 5248, e-mail address schwicht@mathematik.uni-muenchen.de

of polynomial-time computable functions while still admitting higher type recursion, an additional principle is required. It turned out that by introducing linearity constraints in conjunction with ramified recursion, one can characterize polynomial-time computability while admitting recursion in higher types [5, 24]. The resulting restriction **LT** of Gödel's system **T** has as its definable functions exactly the polynomial time computable ones.

In the present paper, we wish to solve the equation

$$\frac{\text{Heyting Arithmetic}}{\text{Gödel's T}} = \frac{???}{\mathbf{LT}}$$

In other words, we seek a logic whose Π^0_2-proofs can be realised by terms in the system **LT** and thus in particular have polynomial-time Skolem functions. To this end we define a restriction **LHA** of Heyting arithmetic **HA** which incorporates linearity as well as ramification. More precisely, we combine

- a liberalized form of linearity for object and assumption variables (allowing multiple use of ground type results) with
- an extension of ramification concepts to all finite types and formulas, by allowing a "computationally irrelevant" universal quantifier $\forall^{\mathsf{nc}} x^\rho A$, and also $A \to B$ and $\forall \bar{x}^\rho A$ as well as $A \multimap B$ and $\forall x^\rho A$, and a corresponding distinction between complete and incomplete (assumption and object) variables.

We will show that the provably recursive functions of **LHA** are exactly the polynomial-time computable ones.

It is hoped that the present approach will be useful for studying program extraction, since it is based on intuitionistic logic formulated with proof terms, via the Curry-Howard correspondence.

Related work

Work related to the underlying term system **LT** [24] has been done by Hofmann [15], who obtained similar results with a very different proof technique. Ramification concepts have been considered much earlier e.g. by Simmons [25], Bellantoni and Cook [3], Leivant and Marion [20, 19], and Pfenning [23]. Notice however that the "tiered" typed λ-calculi of Leivant and Marion [20] depend heavily on different representations of data (as words and as Church-like abstraction terms), which is not necessary in the **LT**-approach.

It is well known that many complexity classes can be characterized by certain restricted systems of arithmetic, for instance bounded arithmetic (cf. Buss [7], and Clote and Takeuti [8]). One should also mention bounded linear logic of Girard, Scedrov and Scott [12], and the so-called light linear logic of Girard [11]. The former differs from what we do here by requiring

explicit bounds. A precise relation to the latter still needs to be clarified. Hofmann's recursive term system from [15] was lifted to a polytime classical modal arithmetic by Bellantoni and Hofmann [4]. The earlier "intrinsic theories" of Leivant [17] followed the tradition of quantifier restrictions in induction.

A quite different approach to proof theoretic characterizations of polynomial-time computable functions is a restriction on the range of quantifiers to "actual terms" Marion's [21], that is constructor terms with variables; this leads to a particularly simple characterization of polynomial-time. A somewhat similar approach (by means of a two-sorted arithmetic) has been worked out by Ostrin and Wainer [22]. Leivant [18] obtained a characterization of polynomial-time computable functions by a restriction of formulas in crucial positions in a proof, rather than by data-tiering.

2 Motivation

To motivate our restrictions let us look at some examples of arithmetical existence proofs exhibiting exponential growth.

2.1 Double use of assumptions

Consider

$$\begin{array}{c} e(1,y) := \mathsf{S}_0(y) \\ e(\mathsf{S}_i(x),y) := e(x,e(x,y)) \end{array} \quad \text{or} \quad \begin{array}{c} e(1) := \mathsf{S}_0 \\ e(\mathsf{S}_i(x)) := e(x) \circ e(x). \end{array}$$

Then $e(x) = \mathsf{S}_0^{(2^{|x|-1})}$, i.e. e grows exponentially. Here is a corresponding existence proof. We have to show

$$\forall x, y \exists z\, |z| = 2^{|x|-1} + |y|.$$

Proof. By induction on x. The base case is obvious. For the step let x be given and assume (IH) $\forall y \exists z\, |z| = 2^{|x|-1} + |y|$. We must show $\forall y \exists z'\, |z'| = 2^{|x|} + |y|$. Given y, construct z' by using (IH) with y to find z_1, and then using (IH) *again*, this time with z_1, to find z'. □

The double use of the ("functional") induction hypothesis clearly is responsible for the exponential growth. Our linearity restriction will exclude such proofs.

2.2 Substitution in function parameters

Consider the iteration functional

$$I(1,f,y) := y, \quad I(\mathsf{S}_i(x),f,y) := f(I(x,f,y)).$$

I can also be written as a binary function, with unary functions as values:

$$I(1, f) := \mathsf{id}. \quad I(\mathsf{S}_i(x), f) := f \circ I(x, f).$$

Then $I(x, f) = f^{(|x|-1)}$; it is considered feasible in our setting. However, substituting the easily definable doubling function d satisfying $|d(x)| = 2|x|$ yields the exponential function $I(x, d) = d^{(|x|-1)}$. (Note that therefore the functional I cannot be definable in the system $\mathbf{PV}^\omega$ of basic feasible functions (cf. [10, 9]), since the latter is closed under substitution). The corresponding proofs of

$$\forall x.\forall y_1 \exists y_2\, |y_2| = 2|y_1| \to \forall z \exists y\, |y| = 2^{|x|-1} + |z| \tag{1}$$

$$\forall y_1 \exists y_2\, |y_2| = 2|y_1| \tag{2}$$

are unproblematic, but we need to forbid applying a cut here.

Our solution is to introduce a ramification concept. (2) is proved by induction on y_1, hence needs a *complete* quantifier: $\forall \bar{y}_1 \exists y_2\, |y_2| = 2|\bar{y}_1|$. We exclude applicability of a cut by our ramification condition, which requires that the "kernel" (or "body") of (1) – which is to be proved by induction on x – does not contain universal subformulas proved by induction.

2.3 Iterated induction

It might seem that our restrictions are so tight that they rule out any form of nested induction. However, this is not true. One can define e.g. (a form of) multiplication on top of addition: First one proves $\forall \bar{x} \forall y \exists z\, |z| = |\bar{x}| + |y|$ by induction on $\bar{x}$, and then $\forall \bar{y} \exists z\, |z| = |\bar{x}| \cdot |\bar{y}|$ by induction on $\bar{y}$ with a parameter $\bar{x}$.

Note that the distinction in Hofmann [16] between iteration and recursion operators does not show up here, since in our ramified setting the recursion variable will be complete and hence can be used many times.

3 Feasible computation with higher types

Our arithmetical system **LHA** will be modelled after a corresponding term system **LT**. We recall some material from [5, 24].

The *types* are $\rho, \sigma ::= \mathbf{U} \mid \mathbf{B} \mid \mathbf{L}(\rho) \mid \rho \multimap \sigma \mid \rho \to \sigma \mid \rho \otimes \sigma \mid \rho \times \sigma$, and the *level* of a type is defined by

$$\begin{aligned} l(\mathbf{U}) &:= 0 & l(\rho \multimap \sigma) &:= \max\{l(\sigma), 1 + l(\rho)\} \\ l(\mathbf{B}) &:= 0 & l(\rho \to \sigma) &:= \max\{l(\sigma), 1 + l(\rho)\} \\ l(\mathbf{L}(\rho)) &:= l(\rho) & l(\rho \otimes \sigma) &:= \max\{l(\rho), l(\sigma)\} \\ & & l(\rho \times \sigma) &:= \max\{l(\rho), l(\sigma)\}. \end{aligned}$$

Ground types are the types of level 0, and a *higher* type is any type of level at least 1. The $\to$-free types are also called *linear* types. In particular, each ground type is linear.

The *constant symbols* are

$$\begin{array}{ll} \mathit{xx}\colon \mathbf{U} & *_\rho\colon \rho \multimap \mathbf{L}(\rho) \multimap \mathbf{L}(\rho) \\ \mathit{tt}\colon \mathbf{B} & \mathsf{if}_\tau\colon \mathbf{B} \multimap \tau \times \tau \multimap \tau \quad (\tau \text{ linear}) \\ \mathit{ff}\colon \mathbf{B} & \boldsymbol{c}^\rho_\tau\colon \mathbf{L}(\rho) \multimap \tau \times (\rho \multimap \mathbf{L}(\rho) \multimap \tau) \multimap \tau \quad (\tau \text{ linear}) \\ \varepsilon_\rho\colon \mathbf{L}(\rho) & \mathcal{R}^\rho_\tau\colon \mathbf{L}(\rho) \to (\rho \to \mathbf{L}(\rho) \to \tau \multimap \tau) \to \tau \multimap \tau \quad (\rho, \tau \text{ linear}) \end{array}$$

and for linear ρ, σ, τ

$$\begin{array}{rl} \otimes^+_{\rho\sigma}\colon & \rho \multimap \sigma \multimap \rho \otimes \sigma \\ \otimes^-_{\rho\sigma\tau}\colon & \rho \otimes \sigma \multimap (\rho \multimap \sigma \multimap \tau) \multimap \tau \\ \times^+_{\rho\sigma}\colon & \rho \multimap \sigma \multimap \rho \times \sigma \qquad (\text{if } \rho, \sigma \text{ ground}) \\ \times^+_{\rho\sigma\tau}\colon & (\tau \multimap \rho) \multimap (\tau \multimap \sigma) \multimap \tau \multimap \rho \times \sigma \qquad (\text{if } l(\rho \times \sigma) > 0) \\ \mathsf{fst}_{\rho\sigma}\colon & \rho \times \sigma \multimap \rho \\ \mathsf{snd}_{\rho\sigma}\colon & \rho \times \sigma \multimap \sigma. \end{array}$$

The restrictions to linear types ρ, σ, τ are needed in the proof of the Normalization theorem in [24]. $\boldsymbol{c}^\rho_\tau$ is used for definition by cases, and $\mathcal{R}^\rho_\tau$ as a recursion operator. Notice that a single recursion operator (over lists) is used here to cover both, numeric and word recursion. The type of $\times^+_{\rho\sigma\tau}$ can be explained as follows. In our linear setting, using a term of type $\rho \times \sigma$ might be allowed only once. So if one component is formed, the other one is lost. Therefore it is perfectly legal to have an occurrence of a higher type incomplete variable in both components. Now the type of $\times^+_{\rho\sigma\tau}$ allows such duplications, via the argument of type τ.

Terms of the form $(\dots(\varepsilon_\rho *_\rho r^\rho_n)\dots *_\rho r^\rho_2) *_\rho r^\rho_1$ are called *lists* (we use reverse infix notation here, writing $l * r$ instead of $*rl$). We will make use of the following abbreviations for $\mathbf{N} := \mathbf{L}(\mathbf{U})$ and $\mathbf{W} := \mathbf{L}(\mathbf{B})$.

$$\begin{array}{ll} 0 := \varepsilon_{\mathbf{U}} & 1 := \varepsilon_{\mathbf{B}} \\ \mathsf{S} := \lambda l^{\mathbf{N}} l * \mathit{xx} & \mathsf{S}_0 := \lambda l^{\mathbf{W}} l * \mathit{ff} \\ & \mathsf{S}_1 := \lambda l^{\mathbf{W}} l * \mathit{tt}. \end{array}$$

Particular lists are $\mathsf{S}(\dots(\mathsf{S}0)\dots)$ and $\mathsf{S}_{i_1}(\dots(\mathsf{S}_{i_n}1)\dots)$. The former are called *unary numerals*, the latter *binary numerals* (or *numerals of type* $\mathbf{W}$).

Definition (LT-terms). **LT** *terms* (terms for short) are built from these constants and typed variables $\bar{x}^\sigma$ (complete variables) and x^σ (incomplete variables) and by introduction and elimination rules for the two type forms $\rho \multimap \sigma$ and $\rho \to \sigma$, i.e.,

c^ρ (constant) |

$\bar{x}^\rho$ (complete variable) |
x^ρ (incomplete variable) |
$(\lambda x^\rho r^\sigma)^{\rho\multimap\sigma}$ |
$(r^{\rho\multimap\sigma}s^\rho)^\sigma$ with higher type incomplete variables in r, s distinct |
$(\lambda \bar{x}^\rho r^\sigma)^{\rho\to\sigma}$ |
$(r^{\rho\to\sigma}s^\rho)^\sigma$ with s complete.

We say that a term is *linear* or *ground* according as its type is. A term s is *complete* if all of its free variables are complete; otherwise it is *incomplete*. By the restriction on incomplete variables in the formation of (rs), every higher type incomplete variable can occur at most once in a given term.

For later use we fix, for every type ρ, a canonically chosen closed term ε^ρ of this type:

$$\begin{array}{ll} \varepsilon^{\mathbf{U}} := \mathsf{xx}, & \varepsilon^{\rho\to\sigma} := \lambda\bar{x}^\rho\varepsilon^\sigma \\ \varepsilon^{\mathbf{B}} := \mathsf{tt}, & \varepsilon^{\rho\multimap\sigma} := \lambda x^\rho\varepsilon^\sigma \\ \varepsilon^{\mathbf{L}(\rho)} := \varepsilon_\rho & \varepsilon^{\rho\otimes\sigma} := \otimes^+_{\rho\sigma}\varepsilon^\rho\varepsilon^\sigma \\ & \varepsilon^{\rho\times\sigma} := \times^+_{\rho\sigma\mathbf{U}}(\lambda x^{\mathbf{U}}\varepsilon^\rho)(\lambda x^{\mathbf{U}}\varepsilon^\sigma)\varepsilon^{\mathbf{U}}. \end{array}$$

The *conversion* rules are as expected: β-conversion (for complete and incomplete variables) plus

$$\begin{array}{lll} \mathsf{if}_\tau\mathsf{tt}s & \mapsto \mathsf{fst}_{\tau\tau}s & \\ \mathsf{if}_\tau\mathsf{ff}s & \mapsto \mathsf{snd}_{\tau\tau}s & \\ \boldsymbol{c}^\rho_\tau\varepsilon_\rho s & \mapsto \mathsf{fst}_{\tau,\sigma}s & \text{for } \sigma := \rho\multimap\mathbf{L}(\rho)\multimap\tau \\ \boldsymbol{c}^\rho_\tau(l *_\rho r)s & \mapsto \mathsf{snd}_{\tau,\sigma}srl & \text{for } \sigma := \rho\multimap\mathbf{L}(\rho)\multimap\tau \\ \mathcal{R}^\rho_\tau\varepsilon_\rho st & \mapsto t & \\ \mathcal{R}^\rho_\tau(l *_\rho r)st & \mapsto srl(\mathcal{R}^\rho_\tau lst) & \\ \otimes^-_{\rho\sigma\tau}(\otimes^+_{\rho\sigma}rs)t & \mapsto trs & \\ \mathsf{fst}_{\rho\sigma}(\times^+_{\rho\sigma}rs) & \mapsto r & \\ \mathsf{snd}_{\rho\sigma}(\times^+_{\rho\sigma}rs) & \mapsto s & \\ \mathsf{fst}_{\rho\sigma}(\times^+_{\rho\sigma\tau}rst) & \mapsto rt & \\ \mathsf{snd}_{\rho\sigma}(\times^+_{\rho\sigma\tau}rst) & \mapsto st. & \end{array}$$

Redexes are subterms shown on the left side of the conversion rules above. We write $r \to r'$ ($r \to^* r'$) is r can be reduced into r' by one (an arbitrary number of) conversion of a subterm.

Notice that projections w.r.t $\rho\otimes\sigma$ can be defined easily: For a term t of type $\rho\otimes\sigma$ we define

$$t0 := \otimes^-_{\rho\sigma\rho}t(\lambda x^\rho\lambda y^\sigma x) \quad \text{and} \quad t1 := \otimes^-_{\rho\sigma\sigma}t(\lambda x^\rho\lambda y^\sigma y).$$

Then clearly

$$(\otimes^+_{\rho\sigma} rs)0 = \otimes^-_{\rho\sigma\rho}(\otimes^+_{\rho\sigma} rs)(\lambda x^\rho \lambda y^\sigma x) \mapsto (\lambda x^\rho \lambda y^\sigma x)rs \to^* r$$
$$(\otimes^+_{\rho\sigma} rs)1 = \otimes^-_{\rho\sigma\sigma}(\otimes^+_{\rho\sigma} rs)(\lambda x^\rho \lambda y^\sigma y) \mapsto (\lambda x^\rho \lambda y^\sigma y)rs \to^* s.$$

A function f is called *definable in* **LT** if there is a closed term $r\colon \mathbf{W} \twoheadrightarrow \ldots \mathbf{W} \twoheadrightarrow \mathbf{W}$ ($\twoheadrightarrow \in \{\to, \multimap\}$) in **LT** denoting this function.

Using a parse dag model of computation, it is shown in [24] that **LT** is closed under reduction, and that the following holds:

Theorem (Normalization). *Let r be a closed* **LT***-term of type* $\mathbf{W} \twoheadrightarrow \ldots \mathbf{W} \twoheadrightarrow \mathbf{W}$ *($\twoheadrightarrow \in \{\to, \multimap\}$). Then r denotes a polytime function.*

The converse is shown in [24] as well:

Lemma (Sufficiency). *Let f be a polynomial-time computable function. Then f is denoted by a closed* **LT***-term r.*

The proof uses a characterization of the polynomial-time computable functions given by Bellantoni and Cook [3]. In section 9 we will give a similar proof, this time via our arithmetical system **LHA**.

4 LHA-Formulas

We assume a given set of predicate symbols $P, Q, \ldots$ of fixed arity ("arity" here means not just the number of arguments, but also covers the type of the arguments.) When writing $P(\vec{r})$ we implicitly assume correct length and types of $\vec{r}$. Moreover, for every type ρ we assume a special predicate symbol $=_\rho$, called equality. The intended interpretation of $=_\rho$ is extensional equality between objects of type ρ.

LHA-*Formulas* (formulas for short) $A, B, \ldots$ are

$$P(\vec{r}) \mid A \to B \mid A \multimap B \mid A \otimes B \mid A \wedge B \mid \forall^{\mathsf{nc}} x^\rho A \mid \forall \bar{x}^\rho A \mid \forall x^\rho A \mid \exists x^\rho A.$$

In $P(\vec{r})$, the $\vec{r}$ are terms from $\mathbf{T}$. Define falsity $\bot$ by $\mathsf{tt} = \mathsf{ff}$ and negation $\neg A$ by $A \multimap \bot$. Disjunction $A \vee B$ can be defined by $\exists x^{\mathbf{B}}.(x = \mathsf{tt} \multimap A) \wedge (x = \mathsf{ff} \multimap B)$. A dot after a quantified variable means that the range of the quantifier extends as far as allowed by the surrounding parentheses.

Implication $A \to B$ is the ordinary one, for multiple uses of the assumption A. In contrast, $A \multimap B$ is the "linear" (or "affine") implication, for at most one use of the hypothesis. The conjunction is the "weak" one corresponding to the pair, i.e. $A_0 \wedge A_1 \multimap A_i$ will be provable, but $(A \multimap B \multimap C) \multimap (A \wedge B \multimap C)$ will not. However, $A \multimap B \multimap C$ and $A \otimes B \multimap C$ will be equivalent.

The quantifier $\forall^{\mathsf{nc}}$ corresponds to the $\{\forall\}$ in Berger's [6] and marks quantification with no computational content, i.e., a proof of $\forall^{\mathsf{nc}} x A$ is of such a

form that the realizing term does not depend on x. When we want computational content, we must either take the "complete" $\forall \bar{x} A$ (for multiple uses of x) or else the "linear" $\forall x A$ (for at most a single use of x).

Every formula A containing the (constructive) existential quantifier can be seen as a "computational problem". We define $\tau(A)$ as the type of a potential realizer of A, i.e. the type of the program to be extracted from a proof of A.

More precisely, we assign to every formula A an object $\tau(A)$ (a type or the symbol ε). In case $\tau(A) = \varepsilon$ proofs of A have no computational content; such formulas A are called *Harrop formulas*, or computationally *irrelevant* (c.i.). Non-Harrop formulas are also called computationally *relevant* (c.r.).

$$\tau(P(\vec{r})) := \varepsilon$$

$$\tau(A \twoheadrightarrow B) := \begin{cases} \tau(B) & \text{if } \tau(A) = \varepsilon \\ \varepsilon & \text{if } \tau(B) = \varepsilon \\ \tau(A) \twoheadrightarrow \tau(B) & \text{otherwise} \end{cases} \quad \text{for } \twoheadrightarrow \in \{\to, \multimap\}$$

$$\tau(A_0 \otimes A_1) := \begin{cases} \tau(A_i) & \text{if } \tau(A_{1-i}) = \varepsilon \\ \tau(A_0) \otimes \tau(A_1) & \text{otherwise} \end{cases}$$

$$\tau(A_0 \wedge A_1) := \begin{cases} \tau(A_i) & \text{if } \tau(A_{1-i}) = \varepsilon \\ \tau(A_0) \times \tau(A_1) & \text{otherwise} \end{cases}$$

$$\tau(\forall^{\mathsf{nc}} x^\rho A) := \tau(A)$$

$$\tau(\forall \bar{x}^\rho A) := \begin{cases} \varepsilon & \text{if } \tau(A) = \varepsilon \\ \rho \to \tau(A) & \text{otherwise} \end{cases}$$

$$\tau(\forall x^\rho A) := \begin{cases} \varepsilon & \text{if } \tau(A) = \varepsilon \\ \rho \multimap \tau(A) & \text{otherwise} \end{cases}$$

$$\tau(\exists x^\rho A) := \begin{cases} \rho & \text{if } \tau(A) = \varepsilon \\ \rho \times \tau(A) & \text{if } l(\rho) = l(\tau(A)) = 0 \\ \rho \otimes \tau(A) & \text{otherwise.} \end{cases}$$

Notice that a formula A is c.i. iff A contains no existential subformula in a strictly positive position.

A formula A is called *linear* if $\tau(A)$ is linear, i.e. $\to$-free. For instance, every formula without complete universal quantifiers $\forall \bar{x}^\rho$ and $\to$ is linear.

5 Proof Terms and Proofs

We consider a formal system of constructive arithmetic; the standard choice for this is Heyting arithmetic **HA** (see e.g. [26]). However, it is convenient here to base our treatment on lists and list induction, rather than on the

(unary) natural numbers and induction on these. **HA** is directly interpretable in this theory, and clearly has the same proof theoretic strength. For convenience we continue to use the name **HA** for our (slightly modified) theory.

Proof terms denote proofs in natural deduction style. Similar to the (object) terms, they are built from assumption constants ("axioms") and complete as well as incomplete assumption and object variables by introduction and elimination rules for $A \to B$, $A \multimap B$, $\forall^{\mathsf{nc}} x^\rho A$, $\forall \bar{x}^\rho A$ and $\forall x^\rho A$.

The axioms can be divided into four groups: induction and cases axioms, logical axioms, equality axioms, and axioms specifying some predicates $P, Q, \ldots$. We will only give the axioms of the first three groups; they define the core system. The last group depends on particular applications.

Axioms are always closed formulas. However, for readability we sometimes omit the leading universal quantifiers. The induction axioms $\mathsf{Ind}_{l,A}$ are, for A and ρ linear

$$\forall \bar{l}^{\mathbf{L}(\rho)}.(\forall \bar{x}^\rho \forall \bar{l}^{\mathbf{L}(\rho)}.A \multimap A[\bar{l} := \bar{l} *_\rho \bar{x}]) \to A[\bar{l}{:=}\varepsilon_\rho] \multimap A.$$

We also provide the cases axioms $\mathsf{Cases}_{l,A}$ and If_A, for A linear:

$$\forall l^{\mathbf{L}(\rho)}.A[l{:=}\varepsilon_\rho] \wedge \forall x^\rho \forall l^{\mathbf{L}(\rho)} A[l := l *_\rho x] \multimap A$$
$$\forall p^{\mathbf{B}}.A[p{:=}\mathsf{tt}] \wedge A[p{:=}\mathsf{ff}] \multimap A.$$

Let $l(\varepsilon) := 0$. Logical axioms:

$$A \multimap B \multimap A \otimes B$$
$$A \otimes B \multimap (A \multimap B \multimap C) \multimap C$$
$$A \multimap B \multimap A \wedge B \qquad \text{if } l(\tau(A)) = l(\tau(B)) = 0$$
$$(C \multimap A) \multimap (C \multimap B) \multimap C \multimap A \wedge B \qquad \text{if } l(\tau(A)) + l(\tau(B)) > 0$$
$$A_0 \wedge A_1 \multimap A_i$$
$$\forall x.A \multimap \exists x A,$$
$$\exists x A \multimap (\forall x.A \multimap B) \multimap B \quad \text{if } x \notin \mathsf{FV}(B)$$
$$\bot \multimap P(x_1, \ldots, x_n).$$

Equality axioms:

Transitivity, symmetry and reflexivity of $=_\rho$

Conversions

$$f = g \multimap x =_\rho y \multimap fx =_\sigma gy$$
$$x_1 = y_1 \multimap \ldots \multimap x_n = y_n \multimap P(x_1, \ldots, x_n) \multimap P(y_1, \ldots, y_n)$$
$$x =_{\mathbf{U}} \mathsf{xx}$$

$\forall x\, fx =_\sigma gx \multimap f = g$
$\mathsf{fst}_{\rho\sigma} z =_\rho \mathsf{fst}_{\rho\sigma} z' \wedge \mathsf{snd}_{\rho\sigma} z =_\sigma \mathsf{snd}_{\rho\sigma} z' \multimap z =_{\rho\times\sigma} z'$
$z0 =_\rho z'0 \to z1 = z'1 \to z =_{\rho\otimes\sigma} z'$.

If we disregard the difference between complete and incomplete variables and also between the two implications $\to$ and $\multimap$, then the axioms are derivable in **HA**.

By an "ordinary proof term" (in **HA**) we mean a standard proof term built from axioms, assumption and object variables by introduction and elimination rules for implication and the universal quantifier:

Definition (Ordinary proof term).

c^A (axiom) |
$\bar{u}^A, u^A$ (complete and incomplete assumption variables) |
$(\lambda \bar{u}^A M^B)^{A\to B} \mid (M^{A\to B} N^A)^B \mid (\lambda u^A M^B)^{A\multimap B} \mid (M^{A\multimap B} N^A)^B \mid$
$(\lambda x^\rho M^A)^{\forall^{\mathsf{nc}} xA} \mid (M^{\forall^{\mathsf{nc}} x^\rho A} r^\rho)^{A[x:=r]} \mid$
$(\lambda \bar{x}^\rho M^A)^{\forall \bar{x}A} \mid (M^{\forall \bar{x}^\rho A} r^\rho)^{A[\bar{x}:=r]} \mid (\lambda x^\rho M^A)^{\forall xA} \mid (M^{\forall x^\rho A} r^\rho)^{A[x:=r]}$.

Here we do not distinguish between $\forall$ and $\forall^{\mathsf{nc}}$, and again disregard the difference between complete and incomplete variables, and the two implications $\to$ and $\multimap$. In the three introduction rules for the universal quantifier we assume the usual condition on free variables, i.e., that x must not be free in the formula of any free assumption variable. In the elimination rules for the universal quantifier, r is a term in **T**.

Every proof term M has a formula A as its type; we shall also speak of a *derivation* M of the formula A.

The proof terms which make up our linear arithmetic will be selected from the ordinary ones, by conditions similar to those that distinguish **LT**-terms from the ordinary terms in **T**. Before we can give this definition, we need to define what an "extracted term" of an ordinary proof term is.

6 Term Extraction

Given a derivation M of a formula A, we define its *extracted term* $[\![M]\!]$, of type $\tau(A)$. This definition is relative to a fixed assignment of object variables to assumption variables: we assign $\bar{x}_{\bar{u}}^{\tau(A)}$ to $\bar{u}^A$, and $x_u^{\tau(A)}$ to u^A.

Definition (Extracted term $[\![M]\!]$). Let M be a derivation of A. If $\tau(A) = \varepsilon$, then $[\![M]\!]$ is defined to be $\varepsilon^{\tau(A)}$, the canonically chosen closed term of this type (see section 3); for the rest of the definition assume $\tau(A) \neq \varepsilon$.

The induction and cases axioms have the extracted terms (for A linear, $\tau := \tau(A)$, and ρ linear in the induction axiom)

$$\begin{aligned}
[\![\mathsf{Ind}_{l,A}]\!] &:= \mathcal{R}^{\mathbf{L}(\rho)}_{\tau} &&: \mathbf{L}(\rho) \to (\rho \to \mathbf{L}(\rho) \to \tau \multimap \tau) \to \tau \multimap \tau \\
[\![\mathsf{Cases}_{l,A}]\!] &:= \boldsymbol{c}_{\tau} &&: \mathbf{L}(\rho) \multimap \tau \times (\rho \multimap \mathbf{L}(\rho) \multimap \tau) \multimap \tau \\
[\![\mathsf{If}_{A}]\!] &:= \mathsf{if}_{\tau} &&: \mathbf{B} \multimap \tau \times \tau \multimap \tau.
\end{aligned}$$

For the other axioms we need to distinguish cases according to which of the formulas involved have computational content. For $\otimes$-introduction we define (writing $[\![A]\!]$ for $[\![c\colon A]\!]$)

$$[\![A \multimap B \multimap A \otimes B]\!] := \begin{cases} \lambda y^{\tau(B)} y & \text{if } \tau(A) = \varepsilon \\ \lambda x^{\tau(A)} x & \text{if } \tau(B) = \varepsilon \\ \otimes^{+}_{\tau(A),\tau(B)} & \text{otherwise} \end{cases}$$

and for $\otimes$-elimination

$$[\![A \otimes B \multimap (A \multimap B \multimap C) \multimap C]\!] := \begin{cases} \lambda z^{\tau(C)} z & \text{if } \tau(A) = \varepsilon,\ \tau(B) = \varepsilon \\ \lambda y^{\tau(B)} \lambda g^{\tau(B) \multimap \tau(C)} . g y & \text{if } \tau(A) = \varepsilon,\ \tau(B) \neq \varepsilon \\ \lambda x^{\tau(A)} \lambda f^{\tau(A) \multimap \tau(C)} . f x & \text{if } \tau(A) \neq \varepsilon,\ \tau(B) = \varepsilon \\ \otimes^{-}_{\tau(A),\tau(B),\tau(C)} & \text{otherwise.} \end{cases}$$

For $\wedge$-introduction we define $[\![A \multimap B \multimap A \wedge B]\!]$ just as $[\![A \multimap B \multimap A \otimes B]\!]$, but with $\times^{+}_{\tau(A),\tau(B)}$ instead of $\otimes^{+}_{\tau(A),\tau(B)}$, and

$$[\![(C \multimap A) \multimap (C \multimap B) \multimap C \multimap A \wedge B]\!] := \begin{cases} \lambda y^{\tau(B)} y & \text{if } \tau(A) = \varepsilon,\ \tau(C) = \varepsilon \\ \lambda x^{\tau(A)} x & \text{if } \tau(B) = \varepsilon,\ \tau(C) = \varepsilon \\ \lambda x^{\tau(A)} \lambda y^{\tau(B)} . \times^{+}_{\tau(A),\tau(B),\mathbf{U}} (\lambda z x)(\lambda z y) \varepsilon^{\mathbf{U}} & \text{if } \tau(A), \tau(B) \neq \varepsilon,\ \tau(C) = \varepsilon \\ \lambda g^{\tau(C) \multimap \tau(B)} g & \text{if } \tau(A) = \varepsilon,\ \tau(C) \neq \varepsilon \\ \lambda f^{\tau(C) \multimap \tau(A)} f & \text{if } \tau(B) = \varepsilon,\ \tau(C) \neq \varepsilon \\ \times^{+}_{\tau(A),\tau(B),\tau(C)} & \text{otherwise} \end{cases}$$

and for $\wedge$-elimination

$$[\![A \wedge B \multimap A]\!] := \begin{cases} \lambda x^{\tau(A)} x & \text{if } \tau(B) = \varepsilon \\ \mathsf{fst}_{\tau(A),\tau(B)} & \text{otherwise} \end{cases}$$

$$[\![A \wedge B \multimap B]\!] := \begin{cases} \lambda y^{\tau(B)} y & \text{if } \tau(A) = \varepsilon \\ \mathsf{snd}_{\tau(A),\tau(B)} & \text{otherwise.} \end{cases}$$

Finally for the $\exists$-axioms we have in case $l(\rho) + l(\tau(B)) > 0$

$$[\![\forall x^\rho.A \multimap \exists x A]\!] := \begin{cases} \lambda x^\rho x & \text{if } \tau(A) = \varepsilon \\ \otimes^+_{\rho,\tau(A)} & \text{otherwise} \end{cases}$$

$$[\![\exists x^\rho A \multimap (\forall x.A \multimap B) \multimap B]\!] := \begin{cases} \lambda x^\rho \lambda f^{\rho \multimap \tau(B)}.fx & \text{if } \tau(A) = \varepsilon \\ \otimes^-_{\rho,\tau(A),\tau(B)} & \text{otherwise,} \end{cases}$$

and in case $l(\rho) = l(\tau(B)) = 0$

$$[\![\forall x^\rho.A \multimap \exists x A]\!] := \begin{cases} \lambda x^\rho x & \text{if } \tau(A) = \varepsilon \\ \times^+_{\rho,\tau(A)} & \text{otherwise} \end{cases}$$

$$[\![\exists x^\rho A \multimap (\forall x^\rho.A \multimap B) \multimap B]\!] := \begin{cases} \lambda x^\rho \lambda f^{\rho \multimap \tau(B)}.fx & \text{if } \tau(A) = \varepsilon \\ \lambda p^{\rho \times \tau(A)} \lambda f^{\rho \multimap \tau(A) \multimap \tau(B)}.f(\mathsf{fst}p)(\mathsf{snd}p) & \text{otherwise} \end{cases}$$

$$[\![\exists x^\rho A \to (\forall \bar{x}^\rho.A \multimap B) \multimap B]\!] := \begin{cases} \lambda \bar{x}^\rho \lambda f^{\rho \to \tau(B)}.f\bar{x} & \text{if } \tau(A) = \varepsilon \\ \lambda \bar{p}^{\rho \times \tau(A)} \lambda f^{\rho \to \tau(A) \multimap \tau(B)}.f(\mathsf{fst}\bar{p})(\mathsf{snd}\bar{p}) & \text{otherwise} \end{cases}$$

For proof terms which are not axioms we define

$$[\![\bar{u}^A]\!] := \bar{x}^{\tau(A)}_{\bar{u}} \quad (\bar{x}^{\tau(A)}_{\bar{u}} \text{ uniquely associated with } \bar{u}^A)$$

$$[\![u^A]\!] := x^{\tau(A)}_u \quad (x^{\tau(A)}_u \text{ uniquely associated with } u^A)$$

$$[\![\lambda \bar{u}^A M]\!] := \begin{cases} [\![M]\!] & \text{if } \tau(A) = \varepsilon \\ \lambda \bar{x}^{\tau(A)}_{\bar{u}} [\![M]\!] & \text{otherwise} \end{cases}$$

$$[\![\lambda u^A M]\!] := \begin{cases} [\![M]\!] & \text{if } \tau(A) = \varepsilon \\ \lambda x^{\tau(A)}_u [\![M]\!] & \text{otherwise} \end{cases}$$

$$[\![M^{A \to B} N]\!] := [\![M^{A \multimap B} N]\!] := \begin{cases} [\![M]\!] & \text{if } \tau(A) = \varepsilon \\ [\![M]\!][\![N]\!] & \text{otherwise} \end{cases}$$

$$[\![(\lambda x^\rho M)^{\forall^{\mathsf{nc}} x A}]\!] := [\![M]\!]$$

$$[\![M^{\forall^{\mathsf{nc}} x A} r]\!] := [\![M]\!] \quad (r \text{ term of } \mathbf{T})$$

$$[\![(\lambda \tilde{x}^\rho M)^{\forall \tilde{x} A}]\!] := \lambda \tilde{x}^\rho [\![M]\!]$$

$$[\![M^{\forall \tilde{x} A} r]\!] := [\![M]\!] r \quad (r \text{ term of } \mathbf{T}),$$

with $\tilde{x}$ a complete or incomplete variable.

The following can be seen easily: Assume that $M\colon A$ is an ordinary proof term whose free object variables are from Γ (complete variables) and Δ (incomplete variables), and whose free assumption variables are from

$\bar{u}_1^{B_1}, \ldots, \bar{u}_n^{B_n}$ and $v_1^{C_1}, \ldots, v_m^{C_m}$. Then its extracted (object) term $[\![M]\!]\colon \tau(A)$ is in $\mathbf{T}$, and its free variables are from $\Gamma, \bar{x}_{\bar{u}_1}, \ldots, \bar{x}_{\bar{u}_n}$ (the complete ones) and $\Delta, x_{v_1}, \ldots, x_{v_m}$ (the incomplete ones).

7 Modified Realizability and Soundness

Intuitively it is rather clear that the extracted term $[\![M^A]\!]$ of a proof term M^A indeed "realizes" the formula A. However, this can and should be made more precise.

We define ordinary (if we disregard the difference between complete and incomplete variables) **HA**-formulas $r\,\underline{\mathbf{mr}}\,A$, where A is an **LHA**-formula and r is a term of type $\tau(A)$.

$$
\begin{aligned}
\varepsilon\,\underline{\mathbf{mr}}\,P(\vec{r}) &:= P(\vec{r})\\
r\,\underline{\mathbf{mr}}\,(A \to B) &:= r\,\underline{\mathbf{mr}}\,(A \multimap B)\\
&:= \begin{cases}
\varepsilon\,\underline{\mathbf{mr}}\,A \;\to\; r\,\underline{\mathbf{mr}}\,B & \text{if } \tau(A) = \varepsilon\\
\forall x.x\,\underline{\mathbf{mr}}\,A \;\to\; \varepsilon\,\underline{\mathbf{mr}}\,B & \text{if } \tau(A) \neq \varepsilon = \tau(B)\\
\forall x.x\,\underline{\mathbf{mr}}\,A \;\to\; rx\,\underline{\mathbf{mr}}\,B & \text{otherwise}
\end{cases}\\
r\,\underline{\mathbf{mr}}\,(A \otimes B) &:= \begin{cases}
\varepsilon\,\underline{\mathbf{mr}}\,A \;\wedge\; r\,\underline{\mathbf{mr}}\,B & \text{if } \tau(A) = \varepsilon\\
r\,\underline{\mathbf{mr}}\,A \;\wedge\; \varepsilon\,\underline{\mathbf{mr}}\,B & \text{if } \tau(B) = \varepsilon\\
r0\,\underline{\mathbf{mr}}\,A \;\wedge\; r1\,\underline{\mathbf{mr}}\,B & \text{otherwise}
\end{cases}\\
r\,\underline{\mathbf{mr}}\,(A \wedge B) &:= \begin{cases}
\varepsilon\,\underline{\mathbf{mr}}\,A \;\wedge\; r\,\underline{\mathbf{mr}}\,B & \text{if } \tau(A) = \varepsilon\\
r\,\underline{\mathbf{mr}}\,A \;\wedge\; \varepsilon\,\underline{\mathbf{mr}}\,B & \text{if } \tau(B) = \varepsilon\\
\mathsf{fst}_{\tau(A)\tau(B)}r\,\underline{\mathbf{mr}}\,A \;\wedge\; \mathsf{snd}_{\tau(A)\tau(B)}r\,\underline{\mathbf{mr}}\,B & \text{otherwise}
\end{cases}\\
r\,\underline{\mathbf{mr}}\,(\forall^{\mathsf{nc}}xA) &:= \begin{cases}
\forall x.\varepsilon\,\underline{\mathbf{mr}}\,A & \text{if } \tau(A) = \varepsilon\\
\forall x.r\,\underline{\mathbf{mr}}\,A & \text{otherwise}
\end{cases}\\
r\,\underline{\mathbf{mr}}\,(\forall \tilde{x}A) &:= \begin{cases}
\forall \tilde{x}.\varepsilon\,\underline{\mathbf{mr}}\,A & \text{if } \tau(A) = \varepsilon\\
\forall \tilde{x}.r\bar{x}\,\underline{\mathbf{mr}}\,A & \text{otherwise}
\end{cases}\\
r\,\underline{\mathbf{mr}}\,(\exists x^{\rho}A) &:= \begin{cases}
\varepsilon\,\underline{\mathbf{mr}}\,A[x := r] & \text{if } \tau(A) = \varepsilon\\
\mathsf{snd}_{\rho\tau(A)}r\,\underline{\mathbf{mr}}\,A[x := \mathsf{fst}_{\rho\tau(A)}r] & \text{if } l(\rho) = l(\tau(A)) = 0\\
r1\,\underline{\mathbf{mr}}\,A[x := r0] & \text{otherwise,}
\end{cases}
\end{aligned}
$$

again with $\tilde{x}$ a complete or incomplete variable. Notice that for an **HA**-formula A without $\exists x$, the formula $r\,\underline{\mathbf{mr}}\,A$ is provably equivalent to A.

Theorem (Soundness). *Assume that $M\colon A$ is an ordinary proof term whose free assumption variables are from $\bar{u}_1^{B_1}, \ldots, \bar{u}_n^{B_n}$ and $v_1^{C_1}, \ldots, v_m^{C_m}$. Then there is an ordinary proof term $\mu(M)\colon ([\![M]\!]\,\underline{\mathbf{mr}}\,A)$ with free assumptions among $\bar{x}_{\bar{u}_i}\,\underline{\mathbf{mr}}\,B_i$ and $x_{v_j}\,\underline{\mathbf{mr}}\,C_j$.*

Proof. By induction on M. The proof is standard (that is, our restrictions for **LHA** play no role here), and can be found e.g. in [26]. □

8 LHA Proof Terms

We now restrict the rules for generating proof terms in a similar way as we did for object terms. The consequence will be that the extracted term actually is in **LT**.

We simultaneously define **LHA** proof terms M and the set $\mathsf{CV}(M)$ of their "computational variables", which in fact will be the set of variables free in $[\![M]\!]$.

Definition (LHA proof terms). **LHA** proof terms M and the set $\mathsf{CV}(M)$ of their computational variables are defined inductively, as follows.

(a) If $\tau(A) = \varepsilon$, then every ordinary proof term M^A is an **LHA** proof term, and $\mathsf{CV}(M) := \emptyset$.

(b) Every assumption constant (axiom) c^A and every complete or incomplete assumption variable $\bar{u}^A$ or u^A is an **LHA** proof term. $\mathsf{CV}(\bar{u}^A) := \{\bar{x}_{\bar{u}}\}$ and $\mathsf{CV}(u^A) := \{x_u\}$.

(c) If M^A is an **LHA** proof term, then so is $(\lambda\bar{u}^A M)^{A\to B}$ and $(\lambda u^A M)^{A\multimap B}$. $\mathsf{CV}(\lambda\bar{u}^A M) = \mathsf{CV}(M) \setminus \{\bar{x}_{\bar{u}}\}$ and $\mathsf{CV}(\lambda u^A M) = \mathsf{CV}(M) \setminus \{x_u\}$.

(d) If $M^{A\to B}$ and N^A are **LHA** proof terms, then so is $(MN)^B$, provided all variables in $\mathsf{CV}(N)$ are complete. $\mathsf{CV}(MN) := \mathsf{CV}(M) \cup \mathsf{CV}(N)$.

(e) If $M^{A\multimap B}$ and N^A are **LHA** proof terms, then so is $(MN)^B$, provided the higher type incomplete variables in $\mathsf{CV}(M)$ and $\mathsf{CV}(N)$ are disjoint. $\mathsf{CV}(MN) := \mathsf{CV}(M) \cup \mathsf{CV}(N)$.

(f) If M^A is an **LHA** proof term, $x \notin \mathsf{FV}(B)$ for every formula B of a free assumption variable in M, and moreover $x \notin \mathsf{CV}(M)$, then $(\lambda x M)^{\forall^{\mathsf{nc}} xA}$ is an **LHA** proof term. $\mathsf{CV}(\lambda x M) := \mathsf{CV}(M)$.

(g) If M^A is an **LHA** proof term, and $\tilde{x} \notin \mathsf{FV}(B)$ for every formula B of a free assumption variable in M, then so is $(\lambda\tilde{x} M)^{\forall\tilde{x}A}$. $\mathsf{CV}(\lambda\tilde{x} M) := \mathsf{CV}(M) \setminus \{\tilde{x}\}$ ($\tilde{x}$ a complete or incomplete variable).

(h) If $M^{\forall^{\mathsf{nc}} xA}$ is an **LHA** proof term and r is a **T**-term, then $(Mr)^{A[x:=r]}$ is an **LHA** proof term. $\mathsf{CV}(Mr) := \mathsf{CV}(M)$.

(i) If $M^{\forall\bar{x}A}$ is an **LHA** proof term and r is a complete **LT**-term, then $(Mr)^{A[\bar{x}:=r]}$ is an **LHA** proof term. $\mathsf{CV}(Mr) := \mathsf{CV}(M) \cup \mathsf{FV}(r)$.

(j) If $M^{\forall xA}$ is an **LHA** proof term and r is an **LT**-term, then $(Mr)^{A[x:=r]}$ is an **LHA** proof term, provided the higher type incomplete variables in $\mathsf{CV}(M)$ are not free in r. $\mathsf{CV}(Mr) := \mathsf{CV}(M) \cup \mathsf{FV}(r)$.

Let us first verify that the computational variables of an **LHA** proof term M are indeed the variables free in $[\![M]\!]$.

Lemma. *For every* **LHA** *proof term* M *we have* $\mathsf{CV}(M) = \mathsf{FV}([\![M]\!])$.

Proof. Induction on M. We may assume that the derived formula has computational content, for otherwise the claim is obvious.

Case $M^{A \multimap B} N^A$ with $\tau(A) \neq \varepsilon$. Then $\mathsf{CV}(MN) = \mathsf{CV}(M) \cup \mathsf{CV}(N) =_{\mathrm{IH}} \mathsf{FV}([\![M]\!]) \cup \mathsf{FV}([\![N]\!]) = \mathsf{FV}([\![M]\!][\![N]\!]) = \mathsf{FV}([\![MN]\!])$.

Case $(\lambda x M)^{\forall^{\mathsf{nc}} xA}$. Then $\mathsf{CV}(\lambda x M) = \mathsf{CV}(M) =_{\mathrm{IH}} \mathsf{FV}([\![M]\!])$. The claim now follows, since by definition $[\![(\lambda x M)^{\forall^{\mathsf{nc}} xA}]\!] = [\![M]\!]$. □

We can now give a simple characterization of **LHA** proof terms, which refers to extracted terms and **LT** and moreover to the notion of an *nc-correct* (ordinary) proof term, which is defined as follows:

Definition (nc-correct proof terms). We again use $\tilde{x}$ for a complete or incomplete variable.

(a) If $\tau(A) = \varepsilon$, then every ordinary proof term M^A is nc-correct.

(b) Every assumption constant (axiom) c^A and every complete or incomplete assumption variable $\bar{u}^A$ or u^A is an nc-correct proof term.

(c) If M^A is nc-correct, then so is $(\lambda \bar{u}^A M)^{A \to B}$ as well as $(\lambda u^A M)^{A \multimap B}$.

(d) If $M^{A \twoheadrightarrow B}$ and N^A are nc-correct ($\twoheadrightarrow \in \{\to, \multimap\}$), then so is $(MN)^B$.

(e) If M^A is nc-correct, $x \notin \mathsf{FV}(B)$ for every $u^B \in \mathsf{FA}(M)$ and moreover $x \notin \mathsf{FV}([\![M]\!])$, then $(\lambda x M)^{\forall^{\mathsf{nc}} xA}$ is nc-correct.

(f) If M^A is nc-correct, and $\tilde{x} \notin \mathsf{FV}(B)$ for every $u^B \in \mathsf{FA}(M)$, then $(\lambda \tilde{x} M)^{\forall \tilde{x} A}$ is nc-correct.

(g) If $M^{\forall^{\mathsf{nc}} xA}$ is nc-correct and r is a **T**-term, then $(Mr)^{A[x:=r]}$ is nc-correct.

(h) If $M^{\forall \tilde{x} A}$ is nc-correct and r is a **T**-term, then $(Mr)^{A[\tilde{x}:=r]}$ is nc-correct.

Theorem (Characterization of LHA proof terms). *An ordinary proof term* M^A *is an* **LHA** *proof term iff* M *is an nc-correct proof term such that* $[\![M]\!] \in \mathbf{LT}$.

Proof. We proceed by induction on M, assuming that M is an ordinary proof term. We can assume $\tau(A) \neq \varepsilon$, for otherwise the claim is obvious.

Case $M^{A \multimap B} N^A$ with $\tau(A) \neq \varepsilon$. The following are equivalent.

- MN is an **LHA** proof term
- M, N are **LHA** proof terms, and the higher type incomplete variables in $\mathsf{CV}(M)$ and $\mathsf{CV}(N)$ are disjoint
- $[\![M]\!]$ and $[\![N]\!]$ are **LT**-terms, and the higher type incomplete variables in $\mathsf{FV}([\![M]\!])$ and $\mathsf{FV}([\![N]\!])$ are disjoint
- $[\![M]\!][\![N]\!]$ $(= [\![MN]\!])$ is an **LT**-term.

Case $(\lambda x M)^{\forall^{\mathsf{nc}} x A}$. Notice that $x \notin \mathsf{FV}(B)$ for every formula B of a free assumption variable in M, since M is an ordinary proof term. The following are equivalent.

$(\lambda x M)^{\forall^{\mathsf{nc}} x A}$ is an **LHA** proof term
M is an **LHA** proof term, and $x \notin \mathsf{CV}(M)$
$[\![M]\!]$ is an **LT**-term, M is nc-correct and $x \notin \mathsf{FV}([\![M]\!])$
$[\![(\lambda x M)^{\forall^{\mathsf{nc}} x A}]\!]$ $(= [\![M]\!])$ is an **LT**-term and $(\lambda x M)^{\forall^{\mathsf{nc}} x A}$ is nc-correct.

The other cases are similar. □

As expected we can derive $A \otimes B \multimap A \wedge B$, but the converse $A \wedge B \multimap A \otimes B$ is derivable only if $l(\tau(A)) = l(\tau(B)) = 0$. In contrast, $A \wedge B \to A \otimes B$ is easily derivable. Moreover, we can derive $(A \multimap B) \multimap (A \to B)$, $\forall^{\mathsf{nc}} x A \multimap \forall x A$ and $\forall x A \multimap \forall \bar{x} A[x := \bar{x}]$. Note that if $\tau(A) = \varepsilon$, then $\forall x\, A \leftrightarrow \forall^{\mathsf{nc}} x\, A$. Hence in this case it doesn't matter which of the universal quantifiers is used.

The natural deduction framework allows a straightforward formalization of proofs in **LHA**. This applies e.g. to the proofs sketched in sections 2.2 and 2.3. Further examples of derivations will be given in section 9.

9 LHA and its Provably Recursive Functions

An n-ary numerical function f is called *provably recursive* in **LHA** if there is a Σ_1-formula $G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z)$ denoting the graph of f, and a derivation M_f in **LHA** of

$$\forall \tilde{x}_1, \ldots, \forall \tilde{x}_n \exists z G_f(\tilde{x}_1, \ldots, \tilde{x}_n, z).$$

Here the $\tilde{x}_i$ denote complete or incomplete variables of type **W**.

Theorem. *A function is provably recursive in* **LHA** *if and only if it is computable in polynomial time.*

Proof. $\Rightarrow$. Let M be a derivation in **LHA** proving a formula of type $\vec{\mathbf{W}}^k \to \vec{\mathbf{W}}^l \multimap \mathbf{W}$. Then $[\![M]\!]$ belongs to **LT**, hence the claim follows from the Normalization Theorem.

$\Leftarrow$. In Bellantoni and Cook [3] the polynomial time computable functions are characterized by a function algebra B based on untyped schemata of safe recursion and safe composition. There every function is written in the form $f(\vec{x};\vec{y})$ where $\vec{x};\vec{y}$ denotes a bookkeeping of those variables $\vec{x}$ that are used in a recursion defining f, and those variables $\vec{y}$ that are not recursed on. We proceed by induction on the definition of $f(x_1,\ldots,x_k;y_1,\ldots,y_l)$ in B, associating to f a Σ^0_1-formula $G_f(\bar{x}_1,\ldots,\bar{x}_k,y_1,\ldots,y_l,z)$ denoting the graph of f, and a derivation M_f in **LHA** of

$$\forall\bar{x}_1,\ldots,\forall\bar{x}_k\forall y_1,\ldots,\forall y_l\exists z G_f(\bar{x}_1,\ldots,\bar{x}_k,y_1,\ldots,y_l,z).$$

If f in B is an initial function $\mathsf{1}$, S_0, S_1, P, conditional C or projection $\pi^{m,n}_i$, then G_f and M_f are easily defined.

If f is defined by *safe composition* in system B, then

$$f(\vec{x};\vec{y}) := g(r_1(\vec{x};),\ldots,r_m(\vec{x};);s_1(\vec{x};\vec{y}),\ldots,s_n(\vec{x};\vec{y})).$$

Using the induction hypothesis to obtain G_g, $G_{\vec{r}}$, $G_{\vec{s}}$ and M_g, $M_{\vec{r}}$ and $M_{\vec{s}}$, define the Σ^0_1-formula G_f and the derivation M_f in the obvious way.

Finally consider the case when f is defined by *safe recursion*

$$\begin{aligned} f(\mathsf{1},\vec{x};\vec{y}) &:= g(\vec{x};\vec{y}) \\ f(\mathsf{S}_i x,\vec{x};\vec{y}) &:= h_i(x,\vec{x};\vec{y},f(x,\vec{x};\vec{y})). \end{aligned}$$

One has G_g, G_{h_0}, G_{h_1} and M_g, M_{h_0}, M_{h_1} by induction hypothesis. Define $G_f(\bar{x},\vec{\bar{x}},\vec{y},z)$ to mean that there is a list l of the same length as $\bar{x}$, whose last element is z, and such that for all $i < \mathsf{len}(l)$ we have $G_{h_j}(\bar{x}\restriction i,\vec{\bar{x}},\vec{y},l_i,l_{i+1})$ (with $j := 0$ if $\bar{x}_i = \mathsf{tt}$ and $j := 1$ if $\bar{x}_i = \mathsf{ff}$), and also $G_g(\vec{\bar{x}},\vec{y},l_0)$. Now fix the complete variables $\vec{\bar{x}}$, and prove

$$\forall\bar{x}\forall\vec{y}\exists z G_f(\bar{x},\vec{\bar{x}},\vec{y},z).$$

by induction on $\bar{x}$ (notice that the induction formula $\forall\vec{y}\exists z G_f(\bar{x},\vec{\bar{x}},\vec{y},z)$ is linear). The base case follows from $\forall\vec{y}\exists z G_g(\vec{\bar{x}},\vec{y},z)$, and for the step we use

$$\forall\bar{x}\forall\vec{y}\forall y_1\exists z_j G_{h_j}(\bar{x},\vec{\bar{x}},\vec{y},y_1,z_j)$$

and argue as follows. Given $\mathsf{S}_j\bar{x}$, the induction hypothesis on $\bar{x}$ yields y_1 with $G_f(\bar{x},\vec{\bar{x}},\vec{y},y_1)$. But then there is a z_j such that $G_{h_j}(\bar{x},\vec{\bar{x}},\vec{y},y_1,z_j)$, which is what we want. Note that the derivation contains no free incomplete (object or assumption) variables any more ($\vec{y}$ is universally quantified). Note also that z_j is used only once, so the derivation is in **LHA**. □

Acknowledgements

This paper builds an arithmetical system corresponding to the term system in [5, 24], as [1] does for Hofmann's "non-size-increasing" term system [16, 2]; it therefore owes a lot to all the researchers involved. Also, I would like to thank Neil Jones, Frank Pfenning, Stefan Schimanski and an anonymous referee for their useful comments on earlier versions of this work.

References

[1] Klaus Aehlig, Ulrich Berger, Martin Hofmann, and Helmut Schwichtenberg, *An arithmetic for non-size-increasing polynomial time computation*, Theoretical Computer Science **318** (2004), no. 1–2, 3–27.

[2] Klaus Aehlig and Helmut Schwichtenberg, *A syntactical analysis of non-size-increasing polynomial time computation*, ACM Transactions of Computational Logic **3** (2002), no. 3, 383–401.

[3] Stephen Bellantoni and Stephen Cook, *A new recursion-theoretic characterization of the polytime functions*, Computational Complexity **2** (1992), 97–110.

[4] Stephen Bellantoni and Martin Hofmann, *A new "feasible" arithmetic*, The Journal of Symbolic Logic **67** (2002), no. 1, 104–116.

[5] Stephen Bellantoni, Karl-Heinz Niggl, and Helmut Schwichtenberg, *Higher type recursion, ramification and polynomial time*, Annals of Pure and Applied Logic **104** (2000), 17–30.

[6] Ulrich Berger, *Program extraction from normalization proofs*, Typed Lambda Calculi and Applications (M. Bezem and J.F. Groote, eds.), LNCS, vol. 664, Springer Verlag, Berlin, Heidelberg, New York, 1993, pp. 91–106.

[7] Samuel R. Buss, *Bounded arithmetic*, Studies in Proof Theory, Lecture Notes, Bibliopolis, Napoli, 1986.

[8] Peter Clote and Gaisi Takeuti, *Bounded arithmetic for NC, ALogTIME, L and NL*, Annals of Pure and Applied Logic **56** (1992), 73–117.

[9] Stephen A. Cook, *Computability and complexity of higher type functions*, Logic from Computer Science, Proceedings of a Workshop held November 13–17, 1989 (Y.N. Moschovakis, ed.), MSRI Publications, no. 21, Springer Verlag, Berlin, Heidelberg, New York, 1992, pp. 51–72.

[10] Stephen A. Cook and Bruce M. Kapron, *Characterizations of the basic feasible functionals of finite type*, Feasible Mathematics (S. Buss and P. Scott, eds.), Birkhäuser, 1990, pp. 71–96.

[11] Jean-Yves Girard, *Light linear logic*, Information and Computation **143** (1998).

[12] Jean-Yves Girard, Andre Scedrov, and Philipp J. Scott, *Bounded linear logic*, Feasible Mathematics (S.R. Buss and Ph.J. Scott, eds.), Birkhäuser, Boston, 1990, pp. 195–209.

[13] Kurt Gödel, *Über eine bisher noch nicht benützte Erweiterung des finiten Standpunkts*, Dialectica **12** (1958), 280–287.

[14] David Hilbert, *Über das Unendliche*, Mathematische Annalen **95** (1925), 161–190.

[15] Martin Hofmann, *Typed lambda calculi for polynomial-time computation*, Habilitation thesis, Mathematisches Institut, TU Darmstadt, Germany. Available under www.dcs.ed.ac.uk/home/mxh/habil.ps.gz, 1998.

[16] ———, *Linear types and non-size-increasing polynomial time computation*, Proceedings 14'th Symposium on Logic in Computer Science (LICS'99), 1999, pp. 464–473.

[17] Daniel Leivant, *Intrinsic theories and computational complexity*, Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, IN, USA, October 1994 (D. Leivant, ed.), LNCS, vol. 960, Springer Verlag, Berlin, Heidelberg, New York, 1995, pp. 177–194.

[18] ———, *Termination proofs and complexity certification*, Proc 4th TACS (Heidelberg) (B.C. Pierce N. Kobayashi, ed.), LNCS, vol. 2215, Springer, 2001, pp. 183–200.

[19] Daniel Leivant and Jean-Yves Marion, *Ramified recurrence and computational complexity IV: Predicative functionals and poly-space*, To appear: Information and Computation.

[20] ———, *Lambda calculus characterization of poly–time*, Typed Lambda Calculi and Applications (M. Bezem and J.F. Groote, eds.), LNCS Vol. 664, 1993, pp. 274–288.

[21] J-Y Marion, *Actual arithmetic and feasibility*, 15th International workshop, Computer Science Logic, CSL'01 (L.Fribourg, ed.), Lecture Notes in Computer Science, vol. 2142, Springer, 2001, pp. 115–139.

[22] Geoffrey Ostrin and Stanley S. Wainer, *Elementary arithmetic*, Annals of Pure and Applied Logic **133** (2005), 275–292.

[23] Frank Pfenning, *Intensionality, extensionality, and proof irrelevance in modal type theory*, Proceedings of the 16th Annual Symposium on Logic in Computer Science (LICS'01) (J. Halpern, ed.), 2001, pp. 221–230.

[24] Helmut Schwichtenberg and Stephen Bellantoni, *Feasible computation with higher types*, Proof and System-Reliability (H. Schwichtenberg and R. Steinbrüggen, eds.), Proceedings NATO Advanced Study Institute, Marktoberdorf, 2001, Kluwer Academic Publisher, 2002, pp. 399–415.

[25] Harold Simmons, *The realm of primitive recursion*, Archive for Mathematical Logic **27** (1988), 177–188.

[26] Anne S. Troelstra (ed.), *Metamathematical investigation of intuitionistic arithmetic and analysis*, Lecture Notes in Mathematics, vol. 344, Springer Verlag, Berlin, Heidelberg, New York, 1973.