An arithmetic for non-size-increasing polynomial-time computation

Klaus Aehlig [a,1] Ulrich Berger [b,2,3] Martin Hofmann [c]
Helmut Schwichtenberg [a,∗,2]

[a] *Universität München, Mathematisches Institut, Theresienstraße. 39, 80333 München, Germany*

[b] *University of Wales Swansea, Department of Computer Science, Singleton Park, Swansea SA2 8PP, UK*

[c] *Universität München, Institut für Informatik, Oettingenstraße 67, 80538 München, Germany*

Abstract

An arithmetical system is presented with the property that from every proof a realizing term can be extracted that is definable in a certain affine linear typed variant of Gödel's T and therefore defines a non-size-increasing polynomial time computable function.

Key words: logic, arithmetic, implicit computational complexity, non-size-increasing polynomial time computation, realizability, higher types, lambda calculus

1 Introduction

There is an increasing interest in recent research in "implicit computational complexity", e.g. by means of global restrictions on simply typed term systems to en-

∗ Corresponding author.
Email addresses: `aehlig@mathematik.uni-muenchen.de` (Klaus Aehlig), `u.berger@swansea.ac.uk` (Ulrich Berger), `mhofmann@mathematik.uni-muenchen.de` (Martin Hofmann), `schwicht@mathematik.uni-muenchen.de` (Helmut Schwichtenberg).

[1] Supported by the Graduiertenkolleg Logik in der Informatik of the Deutsche Forschungsgemeinschaft.

[2] The hospitality of the Mittag-Leffler Institute in the spring of 2001 is gratefully acknowledged.

[3] Supported by the British Engineering and Physical Research Council.

sure computability in polynomial time [2,7,10,1]. One such approach has its roots in a careful study by Caseiro [4] of many examples of natural algorithms, and her formulation of (partially semantic) criteria ensuring computability in polynomial time. The third author identified in [7] an important aspect of this analysis: the role played by non-size-increasing functions. He designed a new (affine linear) term system which can only define non-size-increasing functions, but still allows nested recursion. One important restriction is that the step terms in recursion operators must be closed, since when unfolding the recursion they will be duplicated and hence would violate linearity otherwise. The first and fourth author gave in [1] a proof of the main result of [7] by a different (syntactical) method, which also provides an explicit construction of the bounding polynomials. One motivation for this work was the expectation that the simple approach chosen should make it easy to design a reasonably rich and flexible (higher type) arithmetical system, whose provably recursive functions can be computed in polynomial time. It is the purpose of the present paper to carry this out.

The leading intuition is of course that one should use the Curry-Howard correspondence between terms in lambda-calculus and derivations in arithmetic. However, care is taken to arrive at a flexible and easy to use arithmetical system, which can be understood in its own right.

The paper is structured as follows. In section 2 we present a variant of the linear term system of [7,1] defining non-size-increasing polynomial time functions only. Tailored for these terms is the arithmetic proof calculus introduced in section 3. In order to obtain a flexible and expressive system we included some unusual features: there are two forms of conjunction, $A \otimes B$ and $A \wedge B$, to account for the linear aspects of our logic. We also distinguish (as in [3]) between quantifiers with and without computational content. The former are obtained by relativizing to special "existence predicates" E_ρ. So $\forall x. E_\rho(x) \to \dots$ and $\exists x. E_\rho(x) \otimes \dots$ indicate that x has computational meaning for the extracted program. The possibility to make this distinction is crucial for obtaining reasonable programs. We also split proof contexts into a "passive" and an "active" part (as done by Reynolds in [13] and by Reddy in [12]), where the latter controls the variables free in the realizing terms. A number of examples shows how the system might be used. In particular we sketch a proof that every list can be sorted. The extracted program is the usual formulation of insertion sort in our term system. In section 4 the link between proofs and programs is made precise via a suitable variant of Kreisel's modified realizability. As corollaries to the soundness theorem we obtain a proof that the provably recursive functions of our system are non-size-increasing and polynomial time computable, and some metamathematical results on our arithmetic system.

2 A term system for non-size-increasing polynomial time computation

We introduce a term system similar to the system in [7]. It will play the same role for our arithmetical system as Gödel's T [5] does for Heyting Arithmetic.

2.1 Types and terms

Definition 2.1 (Finite linear types) *Linear types are defined inductively as*

$$\rho, \sigma ::= \mathbf{U} \mid \diamond \mid \mathbf{L}(\rho) \mid \rho \multimap \sigma \mid \rho \otimes \sigma \mid \rho \times \sigma \mid \rho + \sigma.$$

Definition 2.2 (Set model) *In the (naive) set model every type ρ in the left column, below, is interpreted by the set $\mathbb{S}^\rho$ given in the right column:*

$\mathbf{U}$	*a special singleton set*
$\diamond$	*an unspecified nonempty set*
$\mathbf{L}(\rho)$	*the set of lists of elements of $\mathbb{S}^\rho$*
$\rho \multimap \sigma$	*the set of total functions from $\mathbb{S}^\rho$ to $\mathbb{S}^\sigma$*
$\rho \otimes \sigma$ *and* $\rho \times \sigma$	*the cartesian product of $\mathbb{S}^\rho$ and $\mathbb{S}^\sigma$*
$\rho + \sigma$	*the disjoint sum of $\mathbb{S}^\rho$ and $\mathbb{S}^\sigma$*

Remark 2.3 *Common basic data types like the booleans, as well as unary and binary natural numbers can be defined by $\mathbf{B} := \mathbf{U}+\mathbf{U}$, $\mathbf{N} := \mathbf{L}(\mathbf{U})$, $\mathbf{Bin} := \mathbf{L}(\mathbf{B})$.*

The intuition for the special type $\diamond$ is a pointer to free memory, as in [6]. Since there will be no closed terms of this type, it can be used to ensure that terms contain free variables. For example the type $\diamond \multimap \mathbf{N} \multimap \mathbf{N}$ of the successor function together with the linear typing discipline will make sure that the length of (unary) natural numbers and the more technical measure "number of free variables" will coincide.

Although in the naive set model above $\rho \multimap \sigma$ is interpreted as the full function space, computationally it should be viewed as the type of functions from ρ to σ that are linear in the sense that an argument is used at most once. This aspect will become visible in the typing discipline (definition 2.6).

Similarly, the denotationally equal types $\rho \otimes \sigma$ and $\rho \times \sigma$ have different computational interpretations: from a tensor product $\rho \otimes \sigma$ both components can be used once, whereas in the case of an ordinary pair of type $\rho \times \sigma$, the pair itself can be used only once, i.e. one has to choose one component of the pair. Conversely, when forming an element of type $\rho \otimes \sigma$ from elements r^ρ and s^σ we insist that r and s do not share common free variables, whereas for the construction of elements of type $\rho \times \sigma$ no such restriction applies.

Terms are built from variables $x, y, z, \ldots$ and constants c (definition 2.5). Each variable has a type and it is assumed that there are infinitely many variables of each type. The notation x^ρ should express that the variable x has type ρ.

Definition 2.4 (Terms) *The set of terms is inductively defined by*

$$r, s ::= x^\rho \mid c \mid \lambda x^\rho\, r \mid rs \mid r\,\{s\}$$

These terms should be seen as our "raw syntax"; only correctly typed terms (definition 2.6) will be meaningful. The notation $r\,\{s\}$ (for iteration; cf. the conversion rules in definition 2.9, and remark 3.10) is taken from Joachimski and Matthes [9]. The set $\mathrm{FV}(r)$ of free variables of a term r is defined as usual. In particular $\mathrm{FV}(r\,\{s\}) = \mathrm{FV}(r) \cup \mathrm{FV}(s)$. By $r[s/x]$ we denote the usual substitution of every free occurrence of x in r by s (renaming bound variables in r if necessary). Terms that only differ in the naming of bound variables are identified.

Definition 2.5 (Constants) *The constants and their types are*

ε	$: \mathbf{U}$
nil_ρ	$: \mathbf{L}(\rho)$
cons_ρ	$: \diamond \multimap \rho \multimap \mathbf{L}(\rho) \multimap \mathbf{L}(\rho)$
$\otimes^+_{\rho\sigma}$	$: \rho \multimap \sigma \multimap \rho \otimes \sigma$
$\otimes^-_{\rho\sigma\tau}$	$: \rho \otimes \sigma \multimap (\rho \multimap \sigma \multimap \tau) \multimap \tau$
$\times^+_{\rho\sigma\tau}$	$: (\tau \multimap \rho) \multimap (\tau \multimap \sigma) \multimap \tau \multimap \rho \times \sigma$
$\mathsf{fst}_{\rho\sigma}$	$: \rho \times \sigma \multimap \rho$
$\mathsf{snd}_{\rho\sigma}$	$: \rho \times \sigma \multimap \sigma$
$\mathsf{inl}_{\rho\sigma}$	$: \rho \multimap \rho + \sigma$
$\mathsf{inr}_{\rho\sigma}$	$: \sigma \multimap \rho + \sigma$
$+^-_{\rho\sigma\tau}$	$: \rho + \sigma \multimap (\rho \multimap \tau) \times (\sigma \multimap \tau) \multimap \tau$

Definition 2.6 (Typing) *The relation r^ρ, which should be read 'r has type ρ' is inductively defined as follows:*

$$\frac{}{(x^\rho)^\rho} \qquad (\mathrm{Variable})$$

$$\frac{c : \rho}{c^\rho} \qquad (\mathrm{Constant})$$

$$\frac{r^\sigma}{(\lambda x^\rho\, r)^{\rho \multimap \sigma}} \qquad (\multimap^+)$$

$$\frac{r^{\rho\multimap\sigma} \qquad s^{\rho} \qquad \mathrm{FV}(r)\cap\mathrm{FV}(s)=\emptyset}{(rs)^{\sigma}} \qquad (\multimap^-)$$

$$\frac{r^{\mathbf{L}(\rho)} \qquad s^{\diamond\multimap\rho\multimap\tau\multimap\tau} \qquad \mathrm{FV}(s)=\emptyset}{(r\,\{s\})^{\tau\multimap\tau}} \qquad (\mathbf{L}(\rho)^-)$$

Lemma 2.7 *If r^{ρ} and s^{σ} with $(\mathrm{FV}(r)\setminus\{x^{\sigma}\})\cap\mathrm{FV}(s)=\emptyset$, then $r[s/x^{\sigma}]^{\rho}$.*

PROOF. Easy induction on r.

2.2 Reductions

We now define reduction rules on terms. In order to be able to control the effects of iteration we allow conversion of a term $r\,\{s\}$ only if the iteration argument is already calculated, i.e. if r is a list.

Definition 2.8 (Lists) *Terms of the form $\mathsf{cons}_{\rho} d_1^{\diamond} r_1^{\rho}(\ldots(\mathsf{cons}_{\rho} d_n^{\diamond} r_n^{\rho}\mathsf{nil}_{\rho}))$ will be called* lists *(with n entries).*

Definition 2.9 (Conversions) *$\mapsto$ is defined as:*

$$\begin{array}{lll}
(\lambda x r)s & \mapsto r[s/x] & (\beta\text{-conversion}) \\
\otimes^-_{\rho\sigma\tau}\,(\otimes^+_{\rho\sigma} rs)t & \mapsto trs & \\
\mathsf{fst}_{\rho\sigma}(\times^+_{\rho\sigma\tau} rst) & \mapsto rt & \\
\mathsf{snd}_{\rho\sigma}(\times^+_{\rho\sigma\tau} rst) & \mapsto st & \\
+^-_{\rho\sigma\tau}\,(\mathsf{inl}_{\rho\sigma} r)s & \mapsto \mathsf{fst}_{\rho\sigma} sr & \\
+^-_{\rho\sigma\tau}\,(\mathsf{inr}_{\rho\sigma} r)s & \mapsto \mathsf{snd}_{\rho\sigma} sr & \\
\mathsf{nil}_{\rho}\,\{s\}\,t & \mapsto t & \\
\mathsf{cons}_{\rho} d^{\diamond} rl\,\{s\}\,t & \mapsto sd^{\diamond} r(l\,\{s\}\,t) & \text{provided } l \text{ is a list}
\end{array}$$

Notice that the conversion rules are all correct, with respect to the obvious interpretation of terms in the set model 2.2.

Definition 2.10 (Reduction) *The relation $r \longrightarrow r'$ is inductively defined by*

$$\frac{r \mapsto r'}{r \longrightarrow r'} \qquad \frac{r \longrightarrow r'}{rs \longrightarrow r's} \qquad \frac{s \longrightarrow s'}{rs \longrightarrow rs'} \qquad \frac{r \longrightarrow r'}{r\,\{s\} \longrightarrow r'\,\{s\}} \qquad \frac{s \longrightarrow s'}{r\,\{s\} \longrightarrow r\,\{s'\}}$$

This means, to reduce a term we may convert anywhere, except under λ. A term is called normal, *if it cannot be reduced. We will write $\rightarrow^*$ for the reflexive transitive closure of $\rightarrow$.*

Lemma 2.11 (Subject reduction) *If r^ρ and $r \rightarrow s$, then s^ρ and $\mathrm{FV}(s) \subseteq \mathrm{FV}(r)$.*

PROOF. Induction on r shows that only conversions need to be considered. The only non-trivial case is handled in lemma 2.7.

Definition 2.12 (Almost closed terms) *A term is* almost closed *if all its free variables are of type $\diamond$.*

Proposition 2.13 *Every normal, almost closed term of a type as in the left column is of the form given in the right column:*

$\mathbf{U}$	ε
$\diamond$	*variable*
$\mathbf{L}(\rho)$	*list*
$\rho \otimes \sigma$	$\otimes^+ rs$
$\rho \times \sigma$	$\times^+_{\rho,\sigma,\tau} rst$
$\rho + \sigma$	$\mathsf{inl}\, r$ *or* $\mathsf{inr}\, r$
$\rho \multimap \sigma$	λxr *or* $c\vec{r}$ *or* $r\,\{s\}$

PROOF. Induction on the typing.

Definition 2.14 (Projections) *For terms r of type $\rho \otimes \sigma$ we will use the abbreviations $\pi_0(r) := \otimes^-_{\rho\sigma\rho} r\lambda x^\rho, y^\sigma.x$ and $\pi_1(r) := \otimes^-_{\rho\sigma\sigma} r\lambda x^\rho, y^\sigma.y$. Clearly one has $\pi_0(\otimes^+_{\rho\sigma} st) \rightarrow^* s$ and $\pi_1(\otimes^+_{\rho\sigma} st) \rightarrow^* t$.*

Example 2.15 (Predecessor) *Let us use the abbreviations $0 := \mathsf{nil}_{\mathbf{U}}$ and $\mathsf{S}dr := \mathsf{cons}_{\mathbf{U}} d\varepsilon r$ for the zero and the successor operation on the type $\mathbf{N} := \mathbf{L}(\mathbf{U})$. By the letter ν we will denote numerals, i.e., terms of the form $\mathsf{S}d_1(\ldots(\mathsf{S}d_n 0)\ldots)$. Let*

$$\begin{aligned} s_0 &:= \lambda d^\diamond \lambda z^{\mathbf{N}\otimes(\mathbf{N}\multimap\mathbf{N})}.\otimes^+ (\otimes^- z\lambda n\lambda f.fn)\lambda y^{\mathbf{N}}.\mathsf{S}dy \\ t_0 &:= \otimes^+ 0\,\mathsf{id} \end{aligned}$$

Then the conversion rules imply

$$0\,\{s_0\}\,t_0 \quad \mapsto \otimes^+ 0\,\mathsf{id} \tag{1}$$

$$\mathsf{S}d\nu\,\{s_0\}\,t_0 \rightarrow^* \otimes^+ \nu\lambda y.\mathsf{S}dy \tag{2}$$

The latter can be seen easily by induction on ν. Now the predecessor P can be defined by

$$\mathsf{P} := \lambda x^{\mathbf{N}}.\pi_0(x\,\{s_0\}\,t_0).$$

Definition 2.16 (Pairing) *It is easy to define closed terms*

$$\otimes^+_{\vec\rho} : \vec\rho \multimap \rho_1 \otimes \ldots \otimes \rho_n$$
$$\otimes^-_{\vec\rho\tau} : \rho_1 \otimes \ldots \otimes \rho_n \multimap (\vec\rho \multimap \tau) \multimap \tau$$

that behave like the corresponding constant, i.e.

$$\otimes^-_{\vec\rho\tau}(\otimes^+_{\vec\rho} r_1 \ldots r_n)t \to^* tr_1 \ldots r_n$$

Using these we define a closed term

$$\times^+_{\rho\sigma\vec\tau} : (\vec\tau \multimap \rho) \multimap (\vec\tau \multimap \sigma) \multimap \vec\tau \multimap \rho \times \sigma$$

by, setting $\tau := \tau_1 \otimes \ldots \otimes \tau_n$*,*

$$\times^+_{\rho\sigma\vec\tau} \ := \ \lambda f \lambda g \lambda \vec x.\ \times^+_{\rho\sigma\tau} (\lambda z.\ \otimes^-_{\vec\tau\rho}\, zf)(\lambda z.\ \otimes^-_{\vec\tau\sigma}\, zg)(\otimes^+_{\vec\tau} \vec x)$$

such that $\mathsf{fst}_{\rho\sigma}(\times^+_{\rho\sigma\vec\tau} rs\vec t) \to^* r\vec t$ *and* $\mathsf{snd}_{\rho\sigma}(\times^+_{\rho\sigma\vec\tau} rs\vec t) \to^* s\vec t$*. Now we can define a pairing operation*

$$\langle r^\rho, s^\sigma \rangle^{\rho\times\sigma} \ := \ \times^+_{\rho\sigma\vec\tau}(\lambda\vec x.r)(\lambda\vec x.s)\vec x$$

where $\vec x$ *is a list of the free variables common to* r *and* s*. Obviously*

$$\mathsf{fst}_{\rho\sigma}\langle r, s\rangle \to^* r, \quad \mathsf{snd}_{\rho\sigma}\langle r, s\rangle \to^* s.$$

Note that although the terms r *and* s *may have variables in common, in the term* $\langle r, s\rangle$ *every free variable occurs only once.*

2.3 Lengths of reduction chains

Now we show that every almost closed term of appropriate type in the present system denotes a non-size-increasing polynomial time computable function. We adapt the proof in [7,1] by constructing to every such term a polynomial, whose degree is the nesting of $\{.\}$, bounding the number of reduction steps necessary for computing the result.

Definition 2.17 *For every natural number* n *and every term* r *we define natural numbers* $\#_n(r)$ *and* $\vartheta_n(r)$ *by*

$$\#_n(r) := \begin{cases} k & \textit{if } r \textit{ is a list with } k \textit{ entries and } k \le n \\ n & \textit{otherwise} \end{cases}$$

$$\begin{aligned} \vartheta_n(x) := \vartheta_n(c) \quad &:= \quad 1 \\ \vartheta_n(rs) \quad &:= \quad \vartheta_n(r) + \vartheta_n(s) \end{aligned}$$

$$\begin{array}{lll} \vartheta_n(\lambda x r) & := & \vartheta_n(r) + 1 \\ \vartheta_n(r\,\{s\}) & := & \vartheta_n(r) + (\#_n(r) + 1) \cdot \vartheta_n(s) \end{array}$$

Clearly the function mapping n to $\vartheta_n(r)$ is bounded by a polynomial of degree p where p is the nesting of $\{.\}$ in r.

Lemma 2.18 *(a) $\#_n(r) \geq \#_n(r[s/x])$.*
(b) If $r \to r'$ then $\#_n(r) \geq \#_n(r')$.

PROOF. Obvious from the definition of $\#_n(r)$, using the fact that neither substitution nor reduction change the number of entries of a list.

Lemma 2.19 *If r^ρ then $\vartheta_n(r[s/x]) \leq \vartheta_n(r) + \vartheta_n(s)$.*

PROOF. Induction on r using lemma 2.18 (a) and the fact that in a typed term a free variable can have at most one occurrence .

Definition 2.20 *We write $r \to_n r'$ if $r \to r'$ by converting a subterm s of r with $|\mathrm{FV}(s)| \leq n$, where $|\mathrm{FV}(s)|$ is the number of occurrences of free variables in s.*

Lemma 2.21 *If r^ρ and $r \to_n r'$, then $\vartheta_n(r) > \vartheta_n(r')$.*

PROOF. Induction on r.

Case $r \to_n r'$ is a conversion (definition 2.9), i.e. $r \mapsto r'$ where $|\mathrm{FV}(r)| \leq n$.

Only β-conversion and recursion are critical. While the former is taken care of by lemma 2.19, in a conversion $\mathsf{cons}_\rho d^\diamond r l\,\{s\}\,t \mapsto s d^\diamond r (l\,\{s\}\,t)$ the hypothesis $|\mathrm{FV}(\mathsf{cons}_\rho d^\diamond r l\,\{s\}\,t)| \leq n$ is used: Suppose the list l has k entries. Then $k + 1 \leq n$ because due to the typing rules for terms (definition 2.6) a typed list with $k+1$ entries must have at least $k+1$ occurrences of free variables. Consequently $\#_n(\mathsf{cons}_\rho d^\diamond r l) = k+1$ and $\#_n(l) = k$. Therefore

$$\begin{aligned} &\vartheta_n(\mathsf{cons}_\rho d r l\,\{s\}\,t) \\ &= 1 + \vartheta_n(d) + \vartheta_n(r) + \vartheta_n(l) + (k+2) \cdot \vartheta_n(s) + \vartheta_n(t) \\ &> \vartheta_n(s) + \vartheta_n(d) + \vartheta_n(r) + \vartheta_n(l) + (k+1) \cdot \vartheta_n(s) + \vartheta_n(t) \\ &= \vartheta_n(s d r (l\,\{s\}\,t)) \end{aligned}$$

Case $r \to_n r'$, by converting a proper subterm of r. Easy by induction hypothesis, referring in the case $r\,\{s\} \to r'\,\{s\}$, with $r \to r'$, to lemma 2.18 (b).

Corollary 2.22 *If r^ρ then every reduction sequence starting with r has length $\leq \vartheta_N(r)$ where $N := |\mathrm{FV}(r)|$.*

PROOF. Clearly if $r \to r'$ then $r \to_n r'$ for every $n \geq |\mathrm{FV}(r)|$. Therefore the corollary follows from lemma 2.21 and the subject reduction lemma 2.11.

Proposition 2.23 *Let $r^{\rho \multimap \sigma}$ be a typed term with p nestings of $\{.\}$. Then there is a polynomial B of degree $\max\{p, 1\}$ such that for all terms s^{ρ} containing no $\{.\}$ the term rs reduces to normal form in $\leq B(\mathsf{length}(s))$ steps, where $\mathsf{length}(s)$ is the ordinary (syntactical) length of s.*

PROOF. Let $K := \mathsf{length}(r)$ and $L := \mathsf{length}(s)$. Then $|\mathrm{FV}(rs)| \leq K + L$. Therefore, by lemma 2.22, rs normalizes in $\leq \vartheta_{K+L}(rs)$ steps. Because s doesn't contain $\{.\}$ we have $\vartheta_n(s) = L$ for all n. Consequently

$$\vartheta_{K+L}(rs) = \vartheta_{K+L}(r) + L$$

which is bounded by a polynomial in L of degree $\max\{p, 1\}$.

Definition 2.24 (Data types, data objects, non-size-increasing functions) *A type is called* data type *if it is built from $\mathbf{U}$, $\mathbf{L}(.)$, $\otimes$, and $+$ only (examples: $\mathbf{N}$, $\mathbf{Bin}$, $\mathbf{L}(\mathbf{Bin})$). A* data object *of data type τ is an almost closed term w^{τ} in normal form. The* size *of a data object w^{τ} is the natural size of its denotation, which, by proposition 2.13, essentially, i.e. up to a constant depending only on τ, coincides with the syntactical length, $\mathsf{length}(w)$, and also with the number of free variables, $|\mathrm{FV}(w)|$.*

A function f from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$, where $\vec{\tau}, \tau$ are data types, is called non-size-increasing *if there is a number k such that for all data objects $\vec{a}$ of type $\vec{\tau}$ the result $f(\vec{a})$ has a size less than or equal to the sum of the sizes of the a_i plus k.*

Theorem 2.25 *Let $\vec{\tau}, \tau$ be data types and $r^{\vec{\tau} \multimap \tau}$ be an almost closed term with p nestings of $\{.\}$. Then r defines a polynomial time algorithm for a non-size-increasing function from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$ with computation time bounded by a polynomial of degree $\max\{p, 1\}$.*

PROOF. By proposition 2.13 r defines indeed a function from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$. The assertion about the computation time is proved in proposition 2.23. That r is non-size-increasing follows from the already mentioned facts that reduction does not increase the number of free variables of a term, and that the the size of a data object is essentially the number of its free variables.

Remark 2.26 *In [8] it is shown for a similar term calculus that the converse of theorem 2.25 also holds: Every non-size-increasing function from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$ with computation time bounded by a polynomial is definable by an almost closed term.*

3 Linear arithmetic

We now set up a linear arithmetic tailored for the term system introduced in the previous section.

3.1 Formulas

We assume a fixed set of predicate symbols of fixed arity.

When writing $R(\vec{r})$, R a predicate symbol, we implicitly assume correct length and types of $\vec{r}$. However we only assume that the terms in $\vec{r}$ are *weakly typed*, that is, all restrictions on free variables (when typing terms of the form rs or $\{r\}\, s$) are dropped. This relaxation of the typing rules is necessary because of unrestricted substitutions into formulas allowed by the $\forall$-elimination rule (see definition 3.7).

For every type ρ we assume special predicate symbols E_ρ and $=_\rho$, called existence and equality. We sometimes abbreviate $=_\rho(r,s)$ by $r =_\rho s$ or even $r = s$. The intended interpretation of $=_\rho$ is ordinary extensional equality between objects of type ρ and E_ρ is to be interpreted as the set of all objects of type ρ, that is, all objects do exist. Nevertheless, we will refrain from simply stating the formula $E_\rho(x)$ as an axiom, because we want a proof of $E_\rho(t)$ to provide a construction of the object denoted by t. We will postulate the fact that $E_\rho(x)$ always holds only in a context where the construction of x does not matter. This can be expressed by the axiom scheme $(E_\rho(x) \to A) \to A$, where A is an arbitrary computationally irrelevant formula (see definition 3.5).

In the following the letters P, Q range over predicate symbols different from the existence predicates E_ρ (but including equality $=_\rho$).

Definition 3.1 (Formulas) *The set of formulas is defined inductively:*

$$A, B, C ::= P(\vec{r}) \mid E_\rho(r) \mid A \to B \mid A \otimes B \mid A \wedge B \mid A \vee B \mid \forall x^\rho A \mid \exists x^\rho A.$$

Remark 3.2 *We define falsity $\perp$ by $\mathsf{tt} = \mathsf{ff}$, where $\mathsf{tt} := \mathsf{inl}_{\mathbf{U},\mathbf{U}}\varepsilon$ and $\mathsf{ff} := \mathsf{inr}_{\mathbf{U},\mathbf{U}}\varepsilon$. Negation $\neg A$ is an abbreviation for $A \to \perp$ and $r \neq s$ is shorthand for $\neg(r = s)$.*

The conjunction $\wedge$ is the "weak" one corresponding to the ordinary product $\times$, i.e. $A \wedge B \to A$ and $A \wedge B \to B$ will be provable, but $(A \to B \to C) \to (A \wedge B \to C)$ will not.

The quantifiers correspond to the $\{\forall\}$ in [3] (or the "underlined quantification" in [11]) and mean "quantification without computational content", i.e. a proof of $\forall x A$ is of such a form that the realizing term does not depend on x. When we *want computational content, the quantifiers have to be relativized to the existence predicate, i.e. $\forall x.E(x) \to A$ or $\exists x.E(x) \otimes A$.*

Ex falso quodlibet in the form $\bot \to A$ will not be provable in general: we will not have $\bot \to \exists p^{\diamond} E(p)$, since there is no closed term of type $\diamond$. This is also the reason why disjunction $A \vee B$ cannot be defined by $\exists x^{\mathbf{B}}.E(x) \otimes (x = \mathtt{tt} \to A) \wedge (x = \mathtt{ff} \to B)$: from A we could not conclude e.g. $A \vee \exists p^{\diamond} E(p)$.

Definition 3.3 (Computational content) *For a formula A we define the computational content $\tau(A)$, i.e. the type of its potential realizers, by induction on A.*

$$\begin{aligned}
\tau(E_\rho(r)) &:= \rho \\
\tau(P(\vec{r})) &:= \mathbf{U} \\
\tau(A \to B) &:= \tau(A) \multimap \tau(B) \\
\tau(A \otimes B) &:= \tau(A) \otimes \tau(B) \\
\tau(A \wedge B) &:= \tau(A) \times \tau(B) \\
\tau(A \vee B) &:= \tau(A) + \tau(B) \\
\tau(\forall x^\rho A) &:= \tau(A) \\
\tau(\exists x^\rho A) &:= \tau(A)
\end{aligned}$$

Due to the presence of the type $\mathbf{U}$ types may contain some redundancies. For example, $\rho \multimap \mathbf{U}$ denotes a singleton in the set model and could hence be simplified to $\mathbf{U}$. Let us call a type *clean* if it does not contain redundant parts. Hence the base types $\mathbf{U}$ and $\diamond$ are clean, the types $\rho \multimap \sigma$, $\rho \otimes \sigma$, and $\rho \times \sigma$ are clean if their components ρ and σ are both clean and different from $\mathbf{U}$, and the types $\mathbf{L}(\rho)$ and $\rho + \sigma$ are clean if their components ρ and σ are both clean. To every type ρ we define a canonically isomorphic clean type $\mathsf{c}(\rho)$ as follows.

Definition 3.4 (Cleaning of types)

$$\begin{aligned}
\mathsf{c}(\mathbf{U}) &:= \mathbf{U} \\
\mathsf{c}(\diamond) &:= \diamond \\
\mathsf{c}(\rho \multimap \sigma) &:= \begin{cases} \mathbf{U} & \textit{if } \mathsf{c}(\sigma) = \mathbf{U} \\ \mathsf{c}(\sigma) & \textit{if } \mathsf{c}(\rho) = \mathbf{U} \\ \mathsf{c}(\rho) \multimap \mathsf{c}(\sigma) & \textit{otherwise} \end{cases} \\
\mathsf{c}(\rho \otimes \sigma) &:= \begin{cases} \mathsf{c}(\rho) & \textit{if } \mathsf{c}(\sigma) = \mathbf{U} \\ \mathsf{c}(\sigma) & \textit{if } \mathsf{c}(\rho) = \mathbf{U} \\ \mathsf{c}(\rho) \otimes \mathsf{c}(\sigma) & \textit{otherwise} \end{cases} \\
\mathsf{c}(\rho \times \sigma) &:= \begin{cases} \mathsf{c}(\rho) & \textit{if } \mathsf{c}(\sigma) = \mathbf{U} \\ \mathsf{c}(\sigma) & \textit{if } \mathsf{c}(\rho) = \mathbf{U} \\ \mathsf{c}(\rho) \times \mathsf{c}(\sigma) & \textit{otherwise} \end{cases}
\end{aligned}$$

$$\mathsf{c}(\rho + \sigma) := \mathsf{c}(\rho) + \mathsf{c}(\sigma)$$
$$\mathsf{c}(\mathbf{L}(\rho)) := \mathbf{L}(\mathsf{c}(\rho))$$

We set $\tau^{\mathsf{c}}(A) := \mathsf{c}(\tau(A))$

Essentially we are interested in $\tau^{\mathsf{c}}(A)$ only. However, in order to keep cumbersome case distinctions at bay it will be convenient to consider the uncleaned version $\tau(A)$ as well.

Definition 3.5 (Harrop formulas) *We say that a formula A has no computational content if $\tau^{\mathsf{c}}(A) = \mathbf{U}$. Formulas without computational content are also called* Harrop formulas*, or computationally* irrelevant *(*c.i.*), non-Harrop formulas are also called computationally* relevant *(*c.r.*).*

So, a formula is c.i. iff it contains no existence predicate E_ρ with $\mathsf{c}(\rho) \neq \mathbf{U}$ and no disjunction in a strictly positive position.

3.2 Derivations

Proof terms are intended to denote proofs in natural deduction style. They are built up from ordinary terms r, axioms c and assumption variables $u, v, w, \ldots$. Each assumption variables has a formula as type (in the sense of the Curry-Howard correspondence). For each formula there are infinitely many variables of this type. We write u^A or $u\colon A$ to indicate that the variable u has type A.

Definition 3.6 (Raw proof terms)

$$M, N, L ::= u^A \mid c \mid \lambda u^A M \mid \lambda x^\rho M \mid MN \mid Mr \mid M\{N\}$$

Proof contexts are sets of assumption variables. We denote proof contexts by $\Pi, \Gamma, \ldots$, and write Π, Γ for the union $\Pi \cup \Gamma$, expressing that Π and Γ are disjoint. For contexts consisting of one element we also write u^A instead of $\{u^A\}$. Let $\cdot$ denote the empty proof context.

The term system was based on linearity constraints, hence linearity has to be reflected by the arithmetic in order to achieve a realizability result. However, linearity itself would be too a strong restriction since we will often need to instantiate universal formulas to special terms in order to prove that a certain (c.i.) property holds without actually using the variable (in a relevant way). Therefore we have to allow ourselves to keep assumptions in the context that must not be used in a c.r. way. To achieve this we split the context into two parts: one to control correctness and another one to control linearity. This setup also allows (by the rule $(\mathrm{Passification})$ below) to easily reflect the fact that Harrop formulas have no computational meaning and that therefore the proof of a Harrop formula cannot use any assumption in a c.r. way.

A similar phenomenon appears in the area of syntactic control of interference (SCI), cf. Reynolds [13] or Reddy [12]. There, in a function application rs the two phrases r and s should be "independent", i.e. r should not change something s is reading from or writing to, and conversely. One way to guarantee this is to require that r and s do not share common free variables. However, this requirement seems to be too stringent: one e.g. could not write $+xx$. To relax it, Reynolds identified a special class of values called "passive", which never change the state. Free variables denoting passive values can then be shared by r and s.

Following Reddy [12] we write our typing judgments in the form $\Pi \mid \Gamma \vdash M\colon A$, where the context is split into two parts Π and Γ, with Π considered passive. This is to be read as "M denotes a proof of A in the *passive* context Π and the potentially *active* or *linear* context Γ". The active context controls the variables free in the realizing terms.

Definition 3.7 *The relation $\Pi \mid \Gamma \vdash M\colon A$ is inductively defined as follows.*

$$\frac{}{\Pi \mid \Gamma, u^A \vdash u^A\colon A} \quad (\text{Assumption})$$

$$\frac{}{\Pi \mid \Gamma \vdash c^A\colon A} \quad (\text{Axiom})$$

$$\frac{\Pi \mid \Gamma, u^A \vdash M\colon B}{\Pi \mid \Gamma \vdash \lambda u^A M\colon A \to B} \quad (\to^+)$$

$$\frac{\Pi \mid \Gamma_1 \vdash M\colon A \to B \qquad \Pi \mid \Gamma_2 \vdash N\colon A}{\Pi \mid \Gamma_1, \Gamma_2 \vdash MN\colon B} \quad (\to^-)$$

$$\frac{\Pi \mid \Gamma \vdash M\colon A \qquad \mathtt{VarCond}}{\Pi \mid \Gamma \vdash \lambda x^\rho M\colon \forall x^\rho A} \quad (\forall^+)$$

$$\frac{\Pi \mid \Gamma \vdash M\colon \forall x^\rho A \qquad r \textit{ weakly typed by } \rho}{\Pi \mid \Gamma \vdash Mr\colon A[r/x]} \quad (\forall^-)$$

Here `VarCond` is the usual condition on free variables, i.e. that x must not be free in the type of any element of $\Pi \cup \Gamma$.

We add a rule (Passification) *describing the meaning of the active context: it is only needed to prove non-Harrop formulas. Moreover we add a contraction rule, which can be used to contract the passive part of the context.*

$$\frac{\Pi \mid u^B, \Gamma \vdash M\colon A \qquad A \text{ c.i.}}{\Pi, u^B \mid \Gamma \vdash M\colon A} \qquad (\text{Passification})$$

$$\frac{u^A, \Pi \mid v^A, \Gamma \vdash M\colon B}{\Pi \mid v^A, \Gamma \vdash M[v^A/u^A]\colon B} \qquad (\text{Contraction})$$

We call these rules structural. *The last rule concerns induction.*

$$\frac{\Pi \mid \Gamma \vdash N\colon E(t) \qquad \Pi \mid \cdot \vdash M\colon \forall p^\diamond, x^\tau, l.E(p,x) \to A \to A[\mathsf{cons}(p,x,l)/l]}{\Pi \mid \Gamma \vdash N\,\{M\}\colon A[\mathsf{nil}/l] \to A[t/l]} \qquad (\mathbf{L}(\tau)\text{–Ind})$$

Here $E(p,x) \to \dots$ *is short for* $E(p) \to E(x) \to \dots$.

The axioms can be divided into four groups: logical axioms, equality axioms, axioms for existence predicates, and axioms specifying the additional predicates $P, Q, \dots$. We will only give the axioms of the first three groups. They define the core system. The last group depends on particular applications of the system; examples will be given in section 3.4.

Definition 3.8 (Axioms for the core system)

Logical axioms.

$$(C \to A) \to (C \to B) \to C \to A \wedge B \tag{3}$$
$$A_0 \wedge A_1 \to A_i \tag{4}$$
$$A \to B \to A \otimes B \tag{5}$$
$$A \otimes B \to (A \to B \to C) \to C \tag{6}$$
$$A_i \to A_0 \vee A_1 \tag{7}$$
$$(A \to C) \wedge (B \to C) \to A \vee B \to C \tag{8}$$
$$\forall x.A \to \exists x A \tag{9}$$
$$\exists x A \to (\forall x.A \to B) \to B \quad \textit{if } x \notin \mathrm{FV}(B) \tag{10}$$
$$\bot \to P(\vec{r}) \tag{11}$$

Equality axioms.

Transitivity, symmetry and reflexivity of $=_\rho$. (12)

Equations corresponding to the conversion rules 2.9, where in the equation $\mathsf{cons}_\rho d^\diamond rl\,\{s\}\,t = sd^\diamond r(l\,\{s\}\,t)$ *the term* l *can be arbitrary.* (13)

$$f =_{\rho\multimap\sigma} g \to x =_\rho y \to fx =_\sigma gy \tag{14}$$
$$x_1 = y_1 \to \cdots \to x_n = y_n \to P(x_1,\ldots,x_n) \to P(y_1,\ldots,y_n) \tag{15}$$
$$x =_\rho y \to E(x) \to E(y) \tag{16}$$
$$x =_{\mathbf{U}} \varepsilon \tag{17}$$
$$\forall x\, fx =_\sigma gx \to f =_{\rho\multimap\sigma} g \tag{18}$$
$$\mathsf{fst}\, z =_\rho \mathsf{fst}\, z' \wedge \mathsf{snd}\, z =_\sigma \mathsf{snd}\, z' \to z =_{\rho\times\sigma} z' \tag{19}$$

Axioms for existence predicates.

$$E_{\rho\multimap\sigma}(f) \leftrightarrow \forall x.E_\rho(x) \to E_\sigma(fx) \tag{20}$$
$$E_{\rho\times\sigma}(z) \leftrightarrow E_\rho(\mathsf{fst}\, z) \wedge E_\sigma(\mathsf{snd}\, z) \tag{21}$$
$$E(c) \quad \textit{for each of the constructors } \varepsilon, \otimes^+, \mathsf{inl}, \mathsf{inr}, \mathsf{nil}, \mathsf{cons} \tag{22}$$
$$(\forall x^\rho, y^\sigma.E(x,y) \to A[\otimes^+ xy/z]) \to \forall z^{\rho\otimes\sigma}.E(z) \to A \tag{23}$$
$$(\forall x^\rho.E(x) \to A[\mathsf{inl}\, x/z]) \wedge (\forall y^\sigma.E(y) \to A[\mathsf{inr}\, y/z]) \to \forall z^{\rho+\sigma}.E(z) \to A \tag{24}$$
$$(E_\rho(x) \to A) \to A \quad \textit{for every c.i. formula } A \tag{25}$$

We write M^A for M if there are Π and Γ such that $\Pi \mid \Gamma \vdash M\colon A$. Obviously, A is uniquely determined by M. If we are not interested in the proof term we will also write $\Pi \mid \Gamma \vdash A$ to mean that there exists a proof term M such that $\Pi \mid \Gamma \vdash M\colon A$ is derivable.

3.3 Remarks

Remark 3.9 *It is easy to see that the following rules are admissible:*

$$\frac{\Pi \mid \Gamma \vdash A}{\Pi', \Pi \mid \Gamma, \Gamma' \vdash A} \quad (\text{Weakening})$$

$$\frac{\Pi', \Pi \mid \Gamma \vdash A}{\Pi \mid \Gamma, \Pi' \vdash A} \quad (\text{Activation})$$

Remark 3.10 *Our induction rule (*$\mathbf{L}(\tau)-\mathrm{Ind}$*) corresponds to iteration rather than primitive recursion, since for its premise we must prove* $A[\mathsf{cons}(p,x,l)/l]$ *from*

($E(p,x)$ and) A alone, without having access to the previous induction argument l in the form of an $E(l)$-resource. By mimicking the method in [7] one can see that a strengthened induction rule corresponding to primitive recursion, in the form

$$\frac{\Pi \mid \Gamma \vdash E(t) \quad \Pi \mid \cdot \vdash \forall p, x, l.E(p,x) \to A \wedge E(l) \to A[\mathsf{cons}(p,x,l)/l]}{\Pi \mid \Gamma \vdash A[\mathsf{nil}/l] \to A[t/l]} \qquad (\mathbf{L}(\tau)\text{−Ind}^+)$$

is admissible, by invoking ($\mathbf{L}(\tau)$−Ind) with goal formula $A \wedge E(l)$. Its premise can be proved from the given premise using $E(p,x,l) \to E(\mathsf{cons}(p,x,l))$, and from its conclusion

$$\Pi \mid \Gamma \vdash N\,\{M\}\colon A[\mathsf{nil}/l] \wedge E(\mathsf{nil}) \to A[t/l] \wedge E(t)$$

we clearly obtain $A[\mathsf{nil}/l] \to A[t/l]$, using $E(\mathsf{nil})$.

Notice that due to the use of $\wedge$ rather than $\otimes$ we can access either the induction variable or else the previous result, but are not allowed to do both. It is not possible to derive a strengthened induction rule with $\otimes$ instead of $\wedge$.

Remark 3.11 *We list some further useful facts about the system.*

1. *$A \otimes B \to A \wedge B$ is provable in general, but not $A \wedge B \to A \otimes B$. However, for c.i. formulas A, B we can prove $A \wedge B \leftrightarrow A \otimes B$.*
2. *$x =_\rho y \to A(x) \to A(y)$ is provable for all formulas A.*
3. *If $\mathsf{c}(\rho) = \mathbf{U}$ then $x =_\rho y$ is provable.*
4. *The constructors, $\otimes^+$, $\times^+$, inl, inr, nil and cons are injective, and have mutually disjoint ranges. $\otimes^+$ and $\times^+$ are also surjective. That is, the following formulas are provable.*

   $c\vec{x} = c\vec{y} \to \vec{x} = \vec{y}$ *for each constructor c*
   $c\vec{x} \neq c'\vec{x}$ *for each pair of different constructor c, c' of appropriate types.*
   $\forall z^{\rho\otimes\sigma} \exists x, y\,.\, z = \otimes^+ xy$
   $\forall z^{\rho\times\sigma} \exists x, y\,.\, z = \times^+ xy$

5. *The following formulas are provable.*

   $$E_{\rho\otimes\sigma}(z) \leftrightarrow \exists x^\rho \exists y^\sigma . E_\rho(x) \otimes E_\sigma(y) \otimes z = \otimes^+ xy$$
   $$E_{\rho+\sigma}(z) \leftrightarrow (\exists x^\rho . E_\rho(x) \otimes z = \mathsf{inl}\, x) \vee (\exists y^\sigma . E_\sigma(y) \otimes z = \mathsf{inr}\, y)$$
   $$E_{\mathbf{L}(\rho)}(z) \leftrightarrow z = \mathsf{nil} \vee (\exists d^\diamond, x^\rho, x^{\mathbf{L}(\rho)} . E(d,x,y) \otimes z = \mathsf{cons}\, dxy)$$

6. *$E(\vec{x}) \to E(t)$ is provable for every term t which is correctly typed according to definition 2.6 and whose free variables are among $\vec{x}$.*
7. *$\bot \to A$ is provable provided in $\tau(A)$ (or equivalently in $\tau^{\mathsf{c}}(A)$) the type $\diamond$ does not occur strictly positive.*

PROOF. *1. For the underivability statement see corollary 4.8. The rest follows* *directly from the inference rules in definition 3.7.*

2. Easy induction on A.

3. Easy induction on ρ.

4. Note that all formulas to be proven are c.i. Therefore axiom scheme (25) allows us to prove them under the additional assumption that all objects involved exist. But this is easy, using the other existence axioms and our conversion rules, that is, axioms (13).

5. The implications from right to left follow from the axioms (22). The other implications follow from the elimination schemes (23), (24) and induction (which can be viewed as $E_{\mathbf{L}(\tau)}$-elimination). As an example let us assume $E_{\rho+\sigma}(z)$ and prove $(\exists x^\rho.E_\rho(x) \otimes z = \mathsf{inl}\, x) \vee (\exists y^\sigma.E_\sigma(y) \otimes z = \mathsf{inr}\, y)$. By (24) it suffices to prove this for z of the form $\mathsf{inl}\, x'$ where $E(x')$, and also for z of the form $\mathsf{inl}\, y'$ where $E(y')$. But this is obvious.

6. It suffices to prove that all constants exist and that existence is preserved under the formation of $t\{s\}$. For the constructors this follows directly from the axioms concerning the existence predicates. For the other constants and induction one uses the elimination axioms, (23,24) and induction, as well as the conversion rules (13).

7. First one proves the assertion for formulas A of the form $E_\rho(t)$, by induction on ρ. The general case follows by induction on A, using axiom scheme (11).

3.4 Examples

The following examples are intended to demonstrate the flexibility of the system. Some of the system's (inevitable) limitations are expressed by the underivability results in section 4 (e.g. corollary 4.8).

Example 3.12 (Addition) *Assume $\mathsf{Add}(x, y, z)$ expresses $x + y = z$ for natural numbers, for example, via the computational irrelevant axioms*

$$\begin{aligned} &\mathrm{ax}_0\colon\ \forall x.\, \mathsf{Add}(x, 0, x) \\ &\mathrm{ax}_1\colon\ \forall x, y, z, p.\, \mathsf{Add}(x, y, z) \to \mathsf{Add}(x, \mathsf{S}(p, y), \mathsf{S}(p, z)) \end{aligned}$$

We prove

$$\forall x.E_{\mathbf{N}}(x) \to \forall y.E_{\mathbf{N}}(y) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x, y, z)$$

(which will give us, via program extraction (section 4) a polynomial time algorithm for addition). We argue informally. Assume $u_0\colon E_{\mathbf{N}}(x)$ and $u_1\colon E_{\mathbf{N}}(y)$. We have to show $\exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x, y, z)$. Because of the assumption u_1 we can do this by

induction on y. We need to prove base and step, that is,

$$\begin{array}{l}\exists z.\, E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,0,z)\\ \forall p,y.\, E_{\diamond}(p) \to (\exists z.\, E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,y,z)) \to \exists z'.\, E_{\mathbf{N}}(z') \otimes \mathsf{Add}(x,\mathsf{S}(p,y),z')\end{array}$$

For the base we take $z := x$ and use assumption u_0 and axiom ax_0. For the step we assume $v\colon E_{\diamond}(p)$ and also that we have z with $\mathrm{ih}_1\colon E_{\mathbf{N}}(z)$ and $\mathrm{ih}_2\colon \mathsf{Add}(x,y,z)$. We set $z' := \mathsf{S}(p,z)$ and prove $E_{\mathbf{N}}(z')$ using v and ih_1, and $\mathsf{Add}(x,\mathsf{S}(p,y),z')$ using ax_1 and ih_2.

This is a valid proof in our system since every assumption is used only once and the proof of the step is almost closed, that is, does not use any of the assumtions u_0 or u_1. The corresponding derivation term is (using variables with the same types and formulas as in the informal proof)

$$\lambda x, u_0, y, u_1 \,.\, u_1\, \{\mathrm{STEP}\}\, \mathrm{BASE}$$

where $\mathrm{BASE} := \exists^{+} x u_0 (\mathrm{ax}_0 x)$ *and* $\mathrm{STEP} :=$

$$\lambda p, y, v, \mathrm{ih} \,.\, \exists^{-} \mathrm{ih} \lambda z, \mathrm{ih}' \,.\, \otimes^{-} \lambda \mathrm{ih}_1, \mathrm{ih}_2 \,.\, \exists^{+} \mathsf{S}(p,z) (\otimes^{+} (Mpzv\, \mathrm{ih}_1)(\mathrm{ax}_1 xyzp\, \mathrm{ih}_2))$$

with $\mathrm{ih}\colon \exists z.\, E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,y,z)$, $\mathrm{ih}'\colon E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,y,z)$, and a trivial derivation $M\colon \forall p,x \,.\, E(p) \to E(x) \to E(\mathsf{S}(p,x))$ which is easily obtained from axioms about the predicate E.

Notice that we cannot deduce $\forall x.E_{\mathbf{N}}(x) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,x,z)$ (see corollary 4.10), because we would lack one $E_{\mathbf{N}}$ assumption.

Example 3.13 (Recycling of existence) *By induction we can prove*

$$\forall x.E_{\mathbf{N}}(x) \to (x = 0 \vee x \neq 0) \otimes E_{\mathbf{N}}(x).$$

In the base case we use the axiom $E(0)$ and prove the left branch of the disjunction, which is an axiom. The step requires

$$\forall p,x.E_{\diamond}(p) \to E_{\mathbf{N}}(x) \to E_{\mathbf{N}}(\mathsf{S}(p,x)) \otimes \mathsf{S}(p,x) \neq 0,$$

which follows from the axiom $E_{\diamond}(p) \to E_{\mathbf{N}}(x) \to E_{\mathbf{N}}(\mathsf{S}(p,x))$.

Notice that since disjunction is computationally relevant, we cannot establish

$$\forall x.x = 0 \vee x \neq 0,$$

i.e. decidability of equality without an existence assumption (see corollary 4.10). Notice that the more natural statement

$$\forall x.E_{\mathbf{N}}(x) \to x = 0 \vee x \neq 0$$

follows from the one we've shown, but is strictly weaker since it doesn't allow us to "recycle" the information that x "exists".

We can establish

$$\vdash \forall x.E_{\mathbf{N}}(x) \to \forall y.E_{\mathbf{N}}(y) \to (x = y \vee x \neq y) \otimes E_{\mathbf{N}}(x) \otimes E_{\mathbf{N}}(y).$$

Example 3.14 (Sorting) *Assume that we are given a binary relation $\leq$ of arity (ρ, ρ) such that we can prove*

$$\forall x.E_\rho(x) \to \forall y.E_\rho(y) \to (x \leq y \vee x \not\leq y) \otimes E_\rho(x) \otimes E_\rho(y)$$

Furthermore, we assume a ternary relation Ins axiomatized by c.i. axioms such that $\mathsf{Ins}(x, l, l')$ expresses that if l is a sorted list w.r.t. $\leq$, then so is l', and the members of l' are those of l together with x. From the strengthened induction rule ($\mathbf{N}-\mathrm{Ind}^+$) (cf. remark 3.10) we can derive

$$\forall l, x, p.E(l, x, p) \to \exists l'.E(l') \otimes \mathsf{Ins}(x, l, l'), \tag{26}$$

by induction on l. Base: Take $l' = \mathsf{cons}(p, x, \mathsf{nil})$ (using $E(p, x)$). Step: We have $E(y, q)$ and the (strengthened) IH

$$(\forall x, p.E(x, p) \to \exists l'.E(l') \otimes \mathsf{Ins}(x, l, l')) \wedge E(l).$$

We need to show

$$\forall x, p.E(x, p) \to \exists l''.E(l'') \otimes \mathsf{Ins}(x, \mathsf{cons}(q, y, l), l'').$$

So assume $E(x, p)$. Compare x and y without destroying them, i.e. such that after comparison we still have $E(x, y)$. Case $x \leq y$. Take $l'' = \mathsf{cons}(p, x, \mathsf{cons}(q, y, l))$; here we need the right hand part $E(l)$ of the IH, which together with $E(y, q)$ gives us $E(\mathsf{cons}(q, y, l))$. Case $y \leq x$. Using the left hand part of the IH for our x, p gives l' such that $E(l') \otimes \mathsf{Ins}(x, l, l')$. Take $l'' = \mathsf{cons}(q, y, l')$.

From this, we prove that every list can be sorted. Let $\mathsf{Sort}(l, l')$ express that l' is an ordered permutation of l. We want to show

$$\forall l.E_{\mathbf{L}(\rho)}(l) \to \exists l'.E_{\mathbf{L}(\rho)}(l') \otimes \mathsf{Sort}(l, l').$$

Induction on l. In the step case we argue as follows. We have $E(p, x)$ and the IH

$$\exists l'.E_{\mathbf{L}(\rho)}(l') \otimes \mathsf{Sort}(l, l').$$

We need to show

$$\exists l''.E_{\mathbf{L}(\rho)}(l'') \otimes \mathsf{Sort}(\mathsf{cons}(p, x, l), l'').$$

We have an l' such that $E(l') \otimes \mathsf{Sort}(l, l')$. Apply (26) to $l', x.p$. This gives an l'' such that $E(l'') \otimes \mathsf{Ins}(x, l', l'')$. The claim follows from the computationally irrelevant axiom

$$\mathsf{Sort}(l, l') \to \mathsf{Ins}(x, l', l'') \to \mathsf{Sort}(\mathsf{cons}(p, x, l), l'').$$

For the base case we need the computationally irrelevant axiom $\mathsf{Sort}(\mathsf{nil},\mathsf{nil})$.

4 Realizability

4.1 Definition of modified realizability

We now define what it means for a term r to realize a formula A. The intuition of r being a program calculating examples for existential quantifiers is formalized by the (computationally irrelevant) formula $r\,\underline{\mathbf{mr}}\,A$.

Definition 4.1 ($\underline{\mathbf{mr}}$) *By induction on A we define a formula $r\,\underline{\mathbf{mr}}\,A$ for arbitrary $r^{\tau(A)}$.*

$$\begin{array}{ll}
r\,\underline{\mathbf{mr}}\,E_\rho(s) & := (r =_\rho s) \\
r\,\underline{\mathbf{mr}}\,P(\vec{s}) & := P(\vec{s}) \\
r\,\underline{\mathbf{mr}}\,\forall x A & := \forall x. r\,\underline{\mathbf{mr}}\,A \\
r\,\underline{\mathbf{mr}}\,\exists x A & := \exists x. r\,\underline{\mathbf{mr}}\,A \\
r\,\underline{\mathbf{mr}}\,(A \to B) & := \forall x. x\,\underline{\mathbf{mr}}\,A \to rx\,\underline{\mathbf{mr}}\,B \\
r\,\underline{\mathbf{mr}}\,(A \otimes B) & := \pi_0(r)\,\underline{\mathbf{mr}}\,A \,\wedge\, \pi_1(r)\,\underline{\mathbf{mr}}\,B \\
r\,\underline{\mathbf{mr}}\,(A \wedge B) & := \mathsf{fst}\,r\,\underline{\mathbf{mr}}\,A \,\wedge\, \mathsf{snd}\,r\,\underline{\mathbf{mr}}\,B \\
r\,\underline{\mathbf{mr}}\,(A \vee B) & := (\forall x. r = \mathsf{inl}_{\tau(A),\tau(B)}x \to x\,\underline{\mathbf{mr}}\,A) \,\wedge \\
 & \quad\;\; (\forall y. r = \mathsf{inr}_{\tau(A),\tau(B)}y \to y\,\underline{\mathbf{mr}}\,B)
\end{array}$$

Note that $r\,\underline{\mathbf{mr}}\,A$ contains neither E_ρ nor $\otimes$ nor $\vee$.

Proposition 4.2 *(1) If A contains neither existence predicates nor disjunctions, then $r\,\underline{\mathbf{mr}}\,A$ is provably equivalent to A.*
(2) $r\,\underline{\mathbf{mr}}\,\forall x. E(x) \to A$ is provably equivalent to $\forall x. rx\,\underline{\mathbf{mr}}\,A$.
(3) $r\,\underline{\mathbf{mr}}\,\exists x. E(x) \otimes A$ is provably equivalent to $\pi_1(r)\,\underline{\mathbf{mr}}\,A[\pi_0(r)/x]$.

PROOF. Immediate form the definition and the equality axioms.

4.2 Extracted terms

For each variable u^A we choose a unique variable x_{u^A} of type $\tau(A)$ that is sufficiently different to all variables used so far.

Definition 4.3 (Extracted terms) *For a proof M^A we define its extracted term*

$[\![M]\!]$ *by*

$$\begin{array}{ll}
[\![u^A]\!] & := x_u^{\tau(A)} \\
[\![\lambda u^A M^B]\!] & := \lambda x_u^{\tau(A)}\ [\![M]\!] \\
[\![M^{A\to B}N]\!] & := [\![M]\!]\ [\![N]\!] \\
[\![\lambda x^\rho M^A]\!] & := [\![M]\!] \\
[\![M^{\forall x^\rho A} r]\!] & := [\![M]\!] \\
[\![N^{E_{\mathbf{L}(\rho)}(t)}\{M\}]\!] & := [\![N]\!]\ \{[\![M]\!]\}
\end{array}$$

We now define extracted terms for the axioms. We write $[\![A]\!]$ *for* $[\![c\colon A]\!]$*. If* A *is the Efq-axiom (11), or the axiom (25), or one of the equality axioms except (16), then we define* $[\![A]\!] := \varepsilon^{\tau(A)}$*, where for any type* ρ *we let* ε^ρ *be some closed term of type* ρ*. For the remaining axioms we define*

$$\begin{array}{ll}
[\![(C\to A)\to(C\to B)\to C\to A\wedge B]\!] & := \times^+_{\tau(A),\tau(B),\tau(C)} \\
[\![A\wedge B\to A]\!] & := \mathsf{fst}_{\tau(A),\tau(B)} \\
[\![A\wedge B\to B]\!] & := \mathsf{snd}_{\tau(A),\tau(B)} \\
[\![A\to B\to A\otimes B]\!] & := \otimes^+_{\tau(A),\tau(B)} \\
[\![A\otimes B\to(A\to B\to C)\to C]\!] & := \otimes^-_{\tau(A),\tau(B),\tau(C)} \\
[\![A\to A\vee B]\!] & := \mathsf{inl}_{\tau(A),\tau(B)} \\
[\![B\to A\vee B]\!] & := \mathsf{inr}_{\tau(A),\tau(B)} \\
[\![(A\to C)\wedge(B\to C)\to A\vee B\to C]\!] & := \lambda z_1\lambda z_2.\ +^-_{\rho,\sigma,\tau(A)}\ z_2 z_1 \\
[\![\forall x.A\to\exists xA]\!] & := \mathsf{id}_{\tau(A)} \\
[\![\exists xA\to(\forall x.A\to B)\to B]\!] & := \lambda x^{\tau(A)}\lambda f^{\tau(A)\multimap\tau(B)}.fx \\
[\![x=_\rho y\to E(x)\to E(y)]\!] & := \mathsf{id}_\rho \\
[\![E_{\rho\multimap\sigma}(f)\to\forall x.E_\rho(x)\to E_\sigma(fx)]\!] & := \mathsf{id}_{\rho\multimap\sigma} \\
[\![(\forall x.E_\rho(x)\to E_\sigma(fx))\to E_{\rho\multimap\sigma}(f)]\!] & := \mathsf{id}_{\rho\multimap\sigma} \\
[\![E_{\rho\times\sigma}(z)\to E_\rho(z\mathsf{tt})\wedge E_\sigma(z\mathsf{ff})]\!] & := \mathsf{id}_{\rho\times\sigma} \\
[\![E_\rho(z\mathsf{tt})\wedge E_\sigma(z\mathsf{ff})\to E_{\rho\times\sigma}(z)]\!] & := \mathsf{id}_{\rho\times\sigma} \\
[\![E(c)]\!] & := c \quad \textit{for } c=\varepsilon,\otimes^+,\mathsf{inl},\mathsf{inr},\mathsf{nil},\mathsf{cons}
\end{array}$$

$$\begin{array}{l}
[\![(\forall x^\rho,y^\sigma.E(x,y)\to A[\otimes^+xy/z])\to\forall z^{\rho\otimes\sigma}.E(z)\to A]\!] \\
\qquad := \lambda f\lambda z\,.\ \otimes^-\ zf
\end{array}$$

$$\begin{array}{l}
[\![(\forall x^\rho.E(x)\to A[\mathsf{inl}\,x/z])\wedge(\forall y^\sigma.E(y)\to A[\mathsf{inr}\,y/z])\to\forall z^{\rho+\sigma}.E(z)\to A]\!] \\
\qquad := \lambda z_1\lambda z_2\,.\ +^-_{\rho,\sigma,\tau(A)}\ z_2 z_1
\end{array}$$

Depending on applications there may be more axioms. For each such axiom $\mathrm{ax}\colon C$ *one has to chose a term* $[\![\mathrm{ax}]\!]^{\tau(C)}$ *such that* $[\![\mathrm{ax}]\!]\ \underline{\mathbf{mr}}\, C$ *is provable.*

As for the extracted types, also the extracted terms may contain redundant parts which can be removed by an obvious cleaning procedure for terms. Note that the extracted term of a derivation $\Pi \mid \Gamma \vdash M\colon A$ is weakly typed (cf. section 3.1) with type $\tau(A)$ (see theorem 4.7).

Definition 4.4 (Cleaning of terms) *For every variable x^ρ such that $\mathsf{c}(\rho) \neq \mathbf{U}$ we choose a sufficiently different variable $\tilde{x}^{\mathsf{c}(\rho)}$. Relative to this choice we define for every weakly typed term r^ρ a cleaned term $\mathsf{c}(r)$.*

$$\mathsf{c}(r^\rho) := \varepsilon \quad \textit{if}\ \mathsf{c}(\rho) = \mathbf{U}$$

otherwise

$$\begin{aligned}
&\mathsf{c}(x^\rho) := \tilde{x}^{\mathsf{c}(\rho)}\\
&\mathsf{c}(c) \quad \textit{see below}\\
&\mathsf{c}(\lambda x^\rho\, r^\sigma) := \begin{cases} \mathsf{c}(r) & \textit{if}\ \mathsf{c}(\rho) = \mathbf{U}\\ \lambda \tilde{x}^{\mathsf{c}(\rho)}\, \mathsf{c}(r) & \textit{otherwise}\end{cases}\\
&\mathsf{c}(r^{\rho \multimap \sigma} s^\rho) := \begin{cases} \mathsf{c}(r) & \textit{if}\ \mathsf{c}(\rho) = \mathbf{U}\\ \mathsf{c}(r)\mathsf{c}(s) & \textit{otherwise}\end{cases}\\
&\mathsf{c}(r\,\{s\}) := \mathsf{c}(r)\,\{\mathsf{c}(s)\}
\end{aligned}$$

We still have to define $\mathsf{c}(c)$ for constants c^ρ such that $\mathsf{c}(\rho) \neq \mathbf{U}$. Obviously

$$\mathsf{c}(\mathsf{nil}_\rho) := \mathsf{nil}_{\mathsf{c}(\rho)} \quad \textit{and} \quad \mathsf{c}(\mathsf{cons}_\rho) := \begin{cases} \mathsf{S} & \textit{if}\ \mathsf{c}(\rho) = \mathbf{U}\\ \mathsf{cons}_{\mathsf{c}(\rho)} & \textit{otherwise}\end{cases}$$

For the remaining constants the definition of $\mathsf{c}(c)$ is also straightforward, but requires a somewhat tedious case analysis on whether the corresponding type indices are c.i. or not. For example, for $\otimes^-_{\rho\sigma\tau}$, case $\mathsf{c}(\rho) = \mathbf{U} \neq \mathsf{c}(\sigma)$, we have $\mathsf{c}(\otimes^-_{\rho\sigma\tau}) := \lambda x^{\mathsf{c}(\sigma)} \lambda f^{\mathsf{c}(\sigma) \multimap \mathsf{c}(\tau)}.fx$.

Remark 4.5 *It is easy to see that if r is weakly typed and $r \to r'$, then r' is weakly typed and $\mathsf{c}(r) \to^* \mathsf{c}(r')$. Hence for a weakly typed almost closed term r of type $\vec{\tau} \multimap \tau$, where $\vec{\tau}, \tau$ are data types, the terms r and $\mathsf{c}(r)$ essentially define the same function on data types.*

Example 4.6 (Addition, extracted program) *Let us extract a program from the derivation*

$$\lambda x, u_0, y, u_1 \,.\, u_1\,\{\mathrm{STEP}\}\,\mathrm{BASE}$$

of $\forall x.E_{\mathbf{N}}(x) \to \forall y.E_{\mathbf{N}}(y) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,y,z)$ given in example 3.12. We only show cleaned and normalized programs. From the derivation BASE we extract the program x_0. The program extracted from the derivation STEP is the same as the program extracted from $M\colon \forall p, x\,.\,E(p) \to E(x) \to E(\mathsf{S}(p,x))$, *namely $\lambda d, i \,.\, \mathsf{S}(d,i)$. Therefore the program extracted from the complete derivation is $\lambda x_0, x_1 \,.\, x_1 \,\{\lambda d, i \,.\, \mathsf{S}(d,i)\}\, x_0$.*

4.3 Soundness

For a derivation term M we set $[\![M]\!]^{\mathsf{c}} := \mathsf{c}([\![M]\!])$, and for a derivation context Π, $[\![\Pi]\!] := \{x_u^{\tau(A)} \mid u^A \in \Pi\}$ and $[\![\Pi]\!]^{\mathsf{c}} := \{\widetilde{x_u}^{\tau^{\mathsf{c}}(A)} \mid u^A \in \Pi,\ \tau^{\mathsf{c}}(A) \neq \mathbf{U}\}$.

Theorem 4.7 (Soundness of typing) *Assume $\Pi \mid \Gamma \vdash M\colon A$. Then $[\![M]\!]$ is weakly typed with type $\tau(A)$ and $\mathrm{FV}([\![M]\!]) \subseteq [\![\Pi, \Gamma]\!]$. Moreover,*

$$[\![\Gamma]\!]^{\mathsf{c}} \vdash ([\![M]\!]^{\mathsf{c}})^{\tau^{\mathsf{c}}(A)}$$

PROOF. Inspection of the proof rules and the (cleanings of) extracted terms for the c.r. axioms.

Since by proposition 2.13 we have some knowledge of almost closed, normal terms of the different types, we can as a corollary obtain some underivability results.

Corollary 4.8 *Let $\Diamond := \exists p^{\diamond} E(p)$. The following formulas and schemes are underivable:*

$$\begin{aligned}&\bot \to \Diamond\\&\Diamond \to \Diamond \otimes \Diamond\\&(A \to B \to C) \to A \wedge B \to C\end{aligned}$$

PROOF. *Case* $\bot \to \Diamond$. Recall that $\Diamond := \exists p^{\diamond} E(p)$, hence $\tau^{\mathsf{c}}(\bot \to \Diamond) = \diamond$. So if $\bot \to \Diamond$ were derivable, then by soundness of typing we would have a closed term of type $\diamond$, contradicting proposition 2.13 and the fact that every term reduces to a normal form.

Case $\Diamond \to \Diamond \otimes \Diamond$. If this formula were derivable, then by soundness of typing we would have a closed term r of type $\diamond \multimap \diamond \otimes \diamond$. Let p be a variable of type $\diamond$. Then by proposition 2.13 the normal form of rp would be of the form $\otimes^{+} d_0^{\diamond} d_1^{\diamond}$, with normal terms $d_0^{\diamond}, d_1^{\diamond}$. By proposition 2.13 $d_0^{\diamond}, d_1^{\diamond}$ have to be variables, hence distinct. This is the desired contradiction.

Case $(A \to B \to C) \to A \wedge B \to C$. Instantiate A, B by $\Diamond$ and C by $\Diamond \otimes \Diamond$. Then since the premise $\Diamond \to \Diamond \to \Diamond \otimes \Diamond$ and also $\Diamond \to \Diamond \wedge \Diamond$ clearly are derivable, we could also derive $\Diamond \to \Diamond \otimes \Diamond$, which we have just shown to be impossible.

Theorem 4.9 (Soundness) *Assume $\Pi \mid \Gamma \vdash M\colon A$. Then there is a derivation of $[\![M]\!]\ \underline{\mathbf{mr}}\, A$ from assumptions $x_u\, \underline{\mathbf{mr}}\, B$ for $u^B \in \Pi \cup \Gamma$.*

PROOF. By induction on the definition of $\Pi \mid \Gamma \vdash M\colon A$. Only the axioms and induction are of interest. For the Efq-axioms (11) and the equality axioms except (16) the claim is trivial, since they neither contain existence predicates nor disjunctions, and therefore, by proposition 4.2, part 1., their realization is equivalent to themselves. As for the remaining axioms we restrict ourselves to some of the more interesting cases. Note that because the formula $[\![M]\!]\ \underline{\mathbf{mr}}\, A$ is c.i. we may, using axiom (25), assume that all objects involved exist. More precisely, if $[\![M]\!]\ \underline{\mathbf{mr}}\, A$ is of the form $\forall \vec{z}.B$ we may instead prove $\forall \vec{z}.E(\vec{z}) \to B$.

Case $A \otimes B \to (A \to B \to C) \to C$. Assume $z\, \underline{\mathbf{mr}}\, (A \otimes B)$, i.e. that $\pi_0(z)\, \underline{\mathbf{mr}}\, A \wedge \pi_1(z)\, \underline{\mathbf{mr}}\, B$. As indicated above we may assume that z exists. We must show $(\lambda f.zf)\, \underline{\mathbf{mr}}\, ((A \to B \to C) \to C)$. Assume $f\, \underline{\mathbf{mr}}\, (A \to B \to C)$, i.e. $\forall x, y.x\, \underline{\mathbf{mr}}\, A \to y\, \underline{\mathbf{mr}}\, B \to fxy\, \underline{\mathbf{mr}}\, C$. We must show that $zf\, \underline{\mathbf{mr}}\, C$. Using axiom (23), we may also assume that $z = \otimes^+ xy$ for some existing x^ρ, y^σ. Then $\pi_0(z) = x$ and $\pi_1(z) = y$, so $x\, \underline{\mathbf{mr}}\, A \wedge y\, \underline{\mathbf{mr}}\, B$. Now from $zf = \otimes^+ xyf \mapsto fxy$ the claim follows.

Case $A \to A \vee B$. Assume $x\, \underline{\mathbf{mr}}\, A$. We must show $\mathsf{inl}\, x\, \underline{\mathbf{mr}}\, (A \vee B)$, i.e. $\forall x_1.\mathsf{inl}\, x = \mathsf{inl}\, x_1 \to x_1\, \underline{\mathbf{mr}}\, A$ and $\forall y.\mathsf{inl}\, x = \mathsf{inr}\, y \to y\, \underline{\mathbf{mr}}\, A$. The former follows from the injectivity of the constructor inl, and the latter from the disjointness of the ranges of the constructors inl and inr.

Case $(A \to C) \wedge (B \to C) \to A \vee B \to C$. Assume $z_1\, \underline{\mathbf{mr}}\, ((A \to C) \wedge (B \to C))$, i.e. $\mathsf{fst}\, z_1\, \underline{\mathbf{mr}}\, (A \to C) \wedge \mathsf{snd}\, z_1\, \underline{\mathbf{mr}}\, (B \to C)$. Assume further $z_2\, \underline{\mathbf{mr}}\, (A \vee B)$, i.e.

$$(\forall x.z_2 = \mathsf{inl}_{\tau(A),\tau(B)}\, x \to x\, \underline{\mathbf{mr}}\, A) \wedge (\forall y.z_2 = \mathsf{inr}_{\tau(A),\tau(B)}\, y \to y\, \underline{\mathbf{mr}}\, B).$$

We have to show $+^- z_2 z_1\, \underline{\mathbf{mr}}\, C$. Because we may assume that z_2 exists we can use axiom (24) to write z_2, w.l.o.g., as $z_2 = \mathsf{inl}\, x$. It follows $x\, \underline{\mathbf{mr}}\, A$ and subsequently $z_1 \mathsf{tt} x\, \underline{\mathbf{mr}}\, C$. Since $+^- z_2 z_1 = +^-(\mathsf{inl}\, x) z_1 \mapsto z_1 \mathsf{tt} x$ we may conclude $+^- z_2 z_1\, \underline{\mathbf{mr}}\, C$.

Case $(E_\rho(x) \to A) \to A$ where A is c.i. Assume $f\, \underline{\mathbf{mr}}\, E_\rho(x) \to A$, i.e. $\forall y.y = x \to fy\, \underline{\mathbf{mr}}\, A(x)$. This is equivalent to $fx\, \underline{\mathbf{mr}}\, A(x)$, and in turn, by remark 3.11, part 3, equivalent to $\varepsilon\sigma f\, \underline{\mathbf{mr}}\, A(x)$, which is what we have to show.

Case $\mathbf{L}(\tau)-\mathrm{Ind}$. By IH $[\![N]\!]\ \underline{\mathbf{mr}}\, E(t)$, i.e. $[\![N]\!] = t$, and

$$[\![M]\!]\ \underline{\mathbf{mr}}\, \forall p, x, l.E(p, x) \to A \to A[\mathsf{cons}(p, x, l)/l],$$

i.e. by proposition 4.2

$$\forall p, x, l, z^{\tau(A)}.E(p, x) \to z\, \underline{\mathbf{mr}}\, A \to [\![M]\!]\, pxz\, \underline{\mathbf{mr}}\, A[\mathsf{cons}(p, x, l)/l]. \qquad (27)$$

We must show $[\![N]\!]\,\{[\![M]\!]\}\,\underline{\mathbf{mr}}\,(A[\mathsf{nil}/l] \to A[t/l])$. Thanks to axiom (25) we may assume $E(t)$. This allows us to use induction on t to prove

$$[\![N]\!]\,\{[\![M]\!]\}\,\underline{\mathbf{mr}}\,(A[\mathsf{nil}/l] \to A[t/l])$$

Since $\mathsf{nil}\,\{[\![M]\!]\} \mapsto \mathsf{id}$ and $\mathsf{id}\,\underline{\mathbf{mr}}\,(A[\mathsf{nil}/l] \to A[\mathsf{nil}/l])$ by proposition 4.2, it suffices to prove

$$\begin{aligned}&\forall p, x, l.(l\,\{[\![M]\!]\}\,\underline{\mathbf{mr}}\,(A[\mathsf{nil}/l] \to A)) \to\\ &\qquad \mathsf{cons}(p, x, l)\,\{[\![M]\!]\}\,\underline{\mathbf{mr}}\,(A[\mathsf{nil}/l] \to A[\mathsf{cons}(p, x, l)/l]).\end{aligned}$$

Let p, x, l be given and assume

$$\forall z.z\,\underline{\mathbf{mr}}\,A[\mathsf{nil}/l] \to l\,\{[\![M]\!]\}\,z\,\underline{\mathbf{mr}}\,A \tag{28}$$

$$z\,\underline{\mathbf{mr}}\,A[\mathsf{nil}/l] \tag{29}$$

We must show

$$\mathsf{cons}(p, x, l)\,\{[\![M]\!]\}\,z\,\underline{\mathbf{mr}}\,A[\mathsf{cons}(p, x, l)/l]$$

i.e.

$$[\![M]\!]\,px(l\,\{[\![M]\!]\}\,z)\,\underline{\mathbf{mr}}\,A[\mathsf{cons}(p, x, l)/l].$$

This follows from (27) with $l\,\{[\![M]\!]\}\,z$ for z, using (28) and (29).

4.4 Applications

From the soundness theorem 4.9 together with proposition 2.13 we can obtain more underivability results, making use of the set-theoretic model (cf. definition 2.2).

Corollary 4.10 *The following formulas are underivable:*

$$\begin{aligned}&\exists p, p'\, E_{\mathbf{N}}(\mathsf{S}(p, \mathsf{S}(p', 0))),\\ &\forall x.E_{\mathbf{N}}(x) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x, x, z),\\ &\forall x.x = 0 \vee x \neq 0.\end{aligned}$$

PROOF. *Case* $\exists p, p'\, E_{\mathbf{N}}(\mathsf{S}(p, \mathsf{S}(p', 0))) =: A$. If A were derivable, then by the soundness theorem 4.9 we would have a closed term $r^{\mathbf{N}}$ such that $r\,\underline{\mathbf{mr}}\,A$, i.e.

$$\begin{aligned}&\exists p, p'\, r\,\underline{\mathbf{mr}}\,E_{\mathbf{N}}(\mathsf{S}(p, \mathsf{S}(p', 0))),\\ &\exists p, p'\, E_{\mathbf{N}}(\mathsf{S}(p, \mathsf{S}(p', 0))) \otimes r = \mathsf{S}(p, \mathsf{S}(p', 0)).\end{aligned}$$

Because of soundness w.r.t. the set-theoretic interpretation, the value of the closed term r in the model is 2. By proposition 2.13 the normal form of r is a numeral, hence of the form $\mathsf{S}d_0^{\diamond}(\mathsf{S}d_1^{\diamond}0)$. This implies that we would have a closed term of type $\diamond$, contradicting proposition 2.13.

Case $\forall x.E_{\mathbf{N}}(x) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(x,x,z)$. Instantiate this formula with $\mathsf{S}d0$. If the result were derivable, then by the soundness theorem we would have a closed term r of type $\mathbf{N} \multimap \mathbf{N}$ such that

$$\begin{array}{l} r \,\underline{\mathbf{mr}}\, E_{\mathbf{N}}(\mathsf{S}d0) \to \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(\mathsf{S}d0, \mathsf{S}d0, z) \\ \forall x.x \,\underline{\mathbf{mr}}\, E_{\mathbf{N}}(\mathsf{S}d0) \to rx \,\underline{\mathbf{mr}}\, \exists z.E_{\mathbf{N}}(z) \otimes \mathsf{Add}(\mathsf{S}d0, \mathsf{S}d0, z). \end{array}$$

Instantiate this formula with $\mathsf{S}d0$. Then in the set-theoretic model the premise is true, hence also

$$\exists z.r(\mathsf{S}d0) \,\underline{\mathbf{mr}}\, E_{\mathbf{N}}(z) \otimes \mathsf{Add}(\mathsf{S}d0, \mathsf{S}d0, z).$$

Therefore the closed term $r(\mathsf{S}d0)$ has value 2, which is impossible by the argument of the previous case.

Case $\forall x.x = 0 \vee x \neq 0$. If this formula were derivable, then by the soundness theorem it would be realized by a closed term r of type $\mathbf{U} + \mathbf{U}$, i.e.

$$\begin{array}{l} r \,\underline{\mathbf{mr}}\, \forall x.x = 0 \vee x \neq 0, \\ \forall x.r \,\underline{\mathbf{mr}}\, (x = 0 \vee x \neq 0), \\ \forall x.(r = \mathsf{inl}\,\varepsilon \otimes x = 0) \vee (r = \mathsf{inr}\,\varepsilon \otimes x \neq 0). \end{array}$$

By proposition 2.13 r reduces to either $\mathsf{inl}\,\varepsilon$ or $\mathsf{inr}\,\varepsilon$. Therefore in the set-theoretic model we would have either $\forall x x = 0$ or $\forall x x \neq 0$, which is the desired contradiction.

Corollary 4.11 *Let M be an almost closed derivation of*

$$\forall \vec{x}^{\vec{\tau}}.E(\vec{x}) \to \exists y^{\tau}.E(y) \otimes A(\vec{x}, y)$$

($\vec{\tau}, \tau$ data types) where A contains neither existence predicates nor disjunctions. Then $[\![M]\!]^{\mathsf{c}}$ defines a polynomial time algorithm for a non-size-increasing function from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$ satisfying the specification. That is, for every tuple $\vec{w}^{\vec{\tau}}$ of data objects, the term $[\![M]\!]^{\mathsf{c}}\, \vec{w}$ normalizes in polynomial many steps (in the term length of l) to a data object w^{τ} of the same term length (plus a constant depending only on M) such that $A(\vec{w}, w)$ is provable.

PROOF. Proposition 4.2, theorem 4.7, theorem 2.25 and corollary 4.5.

Corollary 4.12 *Let $\forall x^{\rho}.E(x) \to \exists y^{\sigma}.E(y) \otimes A(x,y)$, A as above, be provable (by an almost closed proof). Then $\exists f^{\rho \multimap \sigma}.E(f) \otimes \forall x^{\rho}.E(x) \to A(x, fx)$ is also provable.*

Remark 4.13 *As for theorem 2.25 also for corollary 4.11 a converse holds: Every polynomial time computable non-size-increasing function f from $\mathbb{S}^{\vec{\tau}}$ to $\mathbb{S}^{\tau}$ can be* *extracted from an almost closed derivation of a formula of the form* $\forall\vec{x}^{\vec{\tau}}.E(\vec{x}) \to \exists y^{\tau}.E(y) \otimes A(\vec{x}, y)$*. This follows from remarks 2.26 and 3.11 (6) according to which we can prove* $\forall\vec{x}^{\vec{\tau}}.E(\vec{x}) \to \exists y^{\tau}.E(y) \otimes y = t\vec{x}$ *for some almost closed term* t *defining* f*.*

References

[1] Klaus Aehlig and Helmut Schwichtenberg. A syntactical analysis of non-size-increasing polynomial time computation. In *Proceedings of the 15'th IEEE Symposium on Logic in Computer Science (LICS '00)*, pages 84 – 91, June 2000.

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

[3] Ulrich Berger. Program extraction from normalization proofs. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 91–106. Springer Verlag, Berlin, Heidelberg, New York, 1993.

[4] Voukko-Helena Caseiro. *Equations for Defining Poly-time Functions*. PhD thesis, University of Oslo, 1997. `ftp.ifi.uio.no/pub/vuokko/`.

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

[6] Martin Hofmann. A type system for bounded space and functional in-place update. To appear: Nordic Journal of Programming. An extended abstract has appeared in 'Programming Languages and Systems' (Proc. ESOP 2000), G. Smolka, ed., Springer LNCS, 2000.

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

[8] Martin Hofmann. The strength of non-size-increasing computation. In *Proceedings 'Principles of programming languages' (POPL'02)*, 2002.

[9] Felix Joachimski and Ralph Matthes. Short Proofs of Normalization for the simply-typed λ-calculus, permutative conversions and Gödel's T. *Archive for Mathematical Logic*, 42(1):59–87, 2003.

[10] Daniel Leivant. Intrinsic reasoning about functional programs I. First order theories. To appear in Annals of Pure and Applied Logic.

[11] Jaco van de Pol. *Termination of Higher-order Rewrite Systems*. PhD thesis, Utrecht University, 1996.

[12] Uday Reddy. Global state considered unnecessary. An introduction to object based semantics. *J. Lisp and Symbolic Computation*, 9:7–76, 1996.

[13] John C. Reynolds. Syntactic control of interference. In *ACM Symp. on Princ. of Programming Lang.*, pages 39–46. ACM, 1978.