An Elementary Fragment of Second-Order Lambda Calculus

KLAUS AEHLIG
and
JAN JOHANNSEN
Ludwig-Maximilians-Universität München

A fragment of second-order lambda calculus (System F) is defined that characterizes the elementary recursive functions. Type quantification is restricted to be non-interleaved and stratified, i.e., the types are assigned levels, and a quantified variable can only be instantiated by a type of smaller level, with a slightly liberalized treatment of the level zero.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*computational logic*; *lambda calculus and related systems*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*complexity of proof procedures*

General Terms: Theory

Additional Key Words and Phrases: elementary recursive functions, complexity, lambda calculus, second order logic

1. INTRODUCTION AND RELATED WORK

Machine-independent characterizations of computational complexity classes are at the core of the research area called *Implicit Computational Complexity* which has received a lot of attention recently. The goal is to give natural descriptions of these classes by *conceptual* means, mostly derived from mathematical logic. In particular it is desirable to go without any explicit mention of bounds or ad hoc initial functions.

The second-order, or polymorphic lambda calculus (System F) [Girard 1971; Reynolds 1974] provides a particularly natural framework for this purpose, as all data-types, such as natural numbers, binary words or trees, can be encoded therein without the use of constructors or initial functions. Unfortunately, full system F has a computational strength far beyond anything reasonable in this context: all functions provably total in second-order arithmetic can be defined.

Authors' addresses: Klaus Aehlig, Mathematisches Institut, Ludwig-Maximilians-Universität München, Theresienstraße 39, 80333 München, Germany; Jan Johannsen, Institut für Informatik, Ludwig-Maximilians-Universität München, Oettingenstraße 67, 80538 München.
Klaus Aehlig was supported by the DFG Graduiertenkolleg 301 "*Logik in der Informatik*". Jan Johannsen was partially supported by the DFG Emmy Noether Programme grant Jo 291/2-2.

Recently there have been approaches to define fragments of system F with a weaker computational strength. Altenkirch and Coquand [2001] proposed a fragment characterizing the functions provably recursive in Peano Arithmetic by restricting type abstraction to first-order types in a single variable. Earlier, Leivant [1991] has used stratification of type abstraction to obtain a fragment characterizing the fourth level $\mathfrak{E}_4$ of the Grzegorczyk hierarchy [Grzegorczyk 1953].

Here we give a characterization of the third level $\mathfrak{E}_3$ of the Grzegorczyk hierarchy, that is, the Kalmár elementary recursive functions. In order to achieve this, we use a stratification of type abstractions into only two levels. This alone would give a system in which all definable functions are elementary recursive. However, the class would presumably not be exhausted, as, for example, subtraction seems to be undefinable.

Therefore we use a primitive product type former and allow a quantified variable of the lowest level to be instantiated by a finite product of itself. Note that product types are definable in system F, however using an additional type quantifier and thus disturbing our stratification.

Different restrictions of system F based on linear logic, and characterizing also the elementary recursive functions, as well as polynomial time, were introduced by Girard [1998] and further elaborated by several authors [Asperti and Roversi 2002; Danos and Joinet 2003].

2. DEFINITIONS

The elementary recursive functions are a natural subclass of the primitive recursive functions that was first defined by Kalmár [1943]. A function $f(x, \vec{y})$ is a bounded sum (a bounded product), if it is defined from $g(x, \vec{y})$ by

$$f(x,\vec{y}) = \sum_{i=0}^{x-1} g(i,\vec{y}) \qquad \Big(\text{resp.} \quad f(x,\vec{y}) = \prod_{i=0}^{x-1} g(i,\vec{y}) \ \Big) .$$

The elementary recursive functions are the least class of number-theoretic functions that contains the constant 0, all projections, successor, addition, modified subtraction $x \dot{-} y := \max(x-y, 0)$ and multiplication, and is closed under composition and bounded sums and products.

It is well-known that the elementary recursive functions coincide with the third level $\mathfrak{E}_3$ of the Grzegorczyk hierarchy [Grzegorczyk 1953], and that they coincide as well with the functions computable in time or space bounded by an elementary recursive function (see e.g. [Clote 1999]).

The functions $\lambda n\,.\,2_k(n)$ for $k \in \mathbb{N}$ are inductively defined as follows: $2_0(n) = n$ and $2_{k+1}(n) = 2^{2_k(n)}$. For every fixed k, this function is elementary recursive, but the binary function $\lambda kn\,.\,2_k(n)$ is not: $\lambda k\,.\,2_k(1)$ eventually majorizes every elementary recursive function.

2.1 The system.

We now give a formal definition of our system, by means of a type assignment calculus. So terms are only the terms of the untyped lambda calculus with pairs, i.e., given by the grammar

$$r, s ::= x \mid rs \mid \lambda x\,.\,r \mid \langle r, s\rangle \mid r\mathsf{L} \mid r\mathsf{R}\ ,$$

where x ranges over an infinite set of variables. We define types of level n for a natural number n. However, we will use only the types of level at most 2. Our type variables also come in different levels; let α_n range over variables of level n.

Definition 2.1. The types τ_n of level n and the flat types τ'_0 of level 0 are inductively given by the following grammar:

$$\tau_n := \alpha_n \mid \tau_n \to \tau_n \mid \tau_n \times \tau_n \mid \forall \alpha_k.\tau_k$$
$$\tau'_0 := \alpha_0 \mid \tau'_0 \times \tau'_0$$

where $k < n$ and τ_k does not contain any free variables other than α_k.

Note that this notion of the level of a type differs from the notion commonly used in the literature, so it should more correctly be called *modified level.* However, since the usual notion is not used in the present work, for sake of brevity we just use the term *level* for the modified notion.

Also note that with respect to our notion of the level of a type, there are no closed types of level 0.

2.2 Contexts and Judgments.

A context Γ is a set of pairs $x : \tau$ of variables and types, where the variables occurring in a context have to be distinct. A typing judgment is of the form $\Gamma \vdash r : \tau$ and expresses that r has type τ in the context Γ. The typing rules are:

$$(\text{var})\ \frac{}{\Gamma \vdash x : \tau} \qquad \text{if } x : \tau \text{ occurs in } \Gamma$$

$$(\to I)\ \frac{\Gamma, x : \sigma \vdash r : \rho}{\Gamma \vdash \lambda x\,.\,r : \sigma \to \rho} \qquad (\to E)\ \frac{\Gamma \vdash r : \sigma \to \rho \qquad \Gamma \vdash s : \sigma}{\Gamma \vdash rs : \rho}$$

$$(\times I)\ \frac{\Gamma \vdash r : \rho \qquad \Gamma \vdash s : \sigma}{\Gamma \vdash \langle r, s\rangle : \rho \times \sigma}$$

$$(\times E^1)\ \frac{\Gamma \vdash r : \sigma \times \rho}{\Gamma \vdash r\mathsf{L} : \sigma} \qquad (\times E^2)\ \frac{\Gamma \vdash r : \sigma \times \rho}{\Gamma \vdash r\mathsf{R} : \rho}$$

$$(\forall I)\ \frac{\Gamma \vdash r : \tau_k}{\Gamma \vdash r : \forall \alpha_k.\tau_k} \qquad \text{if } \alpha_k \text{ does not occur free in } \Gamma$$

$$(\forall E^1)\ \frac{\Gamma \vdash r : \forall \alpha_k.\tau_k}{\Gamma \vdash r : \tau_k[\alpha_k := \sigma_\ell]} \qquad \text{if } \ell \leq k \text{ and } \sigma_\ell \text{ is closed}$$

$$(\forall E^2)\ \frac{\Gamma \vdash r : \forall \alpha_0.\tau_0}{\Gamma \vdash r : \tau_0[\alpha_0 := \sigma'_0]} \qquad \text{where } \sigma'_0 \text{ is flat type.}$$

We will tacitly use the obvious fact that $\Gamma \vdash r{:}\tau$ holds only if all the free variables of r are assigned a type in Γ. The rules are formulated in such a way that weakening is admissible. By a simple induction on the derivation one verifies the following.

Proposition 2.2 (Weakening). *If* $\Gamma \subseteq \Gamma'$ *and* $\Gamma \vdash r : \tau$*, then* $\Gamma' \vdash r : \tau$*.*

2.3 Reductions.

Our system is equipped with the usual reductions of lambda calculus with pairs. Let $\rightsquigarrow$ be the reflexive and transitive closure of the reduction given by the compatible closure of the conversions below, i.e., by allowing application of these conversions to arbitrary subterms.

$$\begin{aligned}(\lambda x \,.\, r)s &\mapsto r[x := s] \\ \langle r, s\rangle \mathsf{L} &\mapsto r \\ \langle r, s\rangle \mathsf{R} &\mapsto s \;.\end{aligned}$$

We denote the induced congruence relation by $=_\beta$, i.e., $=_\beta$ is the symmetric and transitive closure of $\rightsquigarrow$. For technical reasons, in some proofs we will also need the notion of $\beta\eta$-equality, denoted by $=_{\beta\eta}$. It is defined like $=_\beta$, but based on the conversions above together with η-conversion

$$\lambda x \,.\, tx \mapsto t$$

with the proviso that x is not free in t.

It is easily verified that our reductions preserve typing.

Proposition 2.3 (Subject Reduction). *If* $\Gamma \vdash r : \tau$ *and* $r \rightsquigarrow r'$, *then* $\Gamma \vdash r' : \tau$.

2.4 Statement of the main result.

For every type τ, we define the type

$$\tau^* := (\tau \to \tau) \to (\tau \to \tau)\,.$$

For a natural number n, the Church numeral $\underline{n}$ is $\lambda f x \,.\, f^n x$, it can have type τ^* for every τ. The types of natural numbers are $\mathsf{Nat}_0 := \forall\alpha_0.{\alpha_0}^*$ and $\mathsf{Nat}_1 := \forall\alpha_1.{\alpha_1}^*$. It can be shown that the only closed normal inhabitants of the types Nat_i are the Church numerals, and the identity combinator $\mathsf{id} := \lambda x \,.\, x$, which is equivalent to the numeral $\underline{1}$ under η-conversion.

A function $f \colon \mathbb{N}^k \to \mathbb{N}$ is *representable*, if there is a term t_f such that $\vdash t_f : \mathsf{Nat}_1^k \to \mathsf{Nat}_0$ and for all $\vec{n} \in \mathbb{N}^k$, it holds that $t_f \underline{\vec{n}} =_\beta \underline{f(\vec{n})}$. We shall prove below, as Corollary 4.16, that the representable functions are exactly the elementary recursive functions.

2.5 Notation.

As usual, lists of notations for terms, numbers etc. that only differ in successive indices are denoted by leaving out the indices and putting an arrow over the notation. It is usually obvious where to add the missing indices, otherwise we add dots wherever an index is left out. We use one dot if the index runs with the innermost arrow, two dots if the index runs with the next innermost arrow etc., so that e.g. the expression

$$\overrightarrow{t_{.}\overrightarrow{n_{..,.}}}$$

stands for a sequence of the form

$$t_1 n_{1,1} \ldots n_{1,k_1} \;,\; \ldots \;,\; t_r n_{r,1} \ldots n_{r,k_r} \;.$$

3. COMPLETENESS

In this section we show one direction of our claim, namely we show that every elementary recursive function can be represented by a term. To start, it is easy to check that the usual basic arithmetic functions can have the following types

$$\begin{array}{lcll} \mathsf{suc} & := & \lambda nsz\,.\,s(nsz) & : \tau^* \to \tau^* \\ \mathsf{add} & := & \lambda mnsz\,.\,ms(nsz) & : \tau^* \to \tau^* \to \tau^* \\ \mathsf{mult} & := & \lambda mns\,.\,m(ns) & : \tau^* \to \tau^* \to \tau^* \end{array}$$

for every τ. We use these to program a downward typecast, that is a function

$$\downarrow\ :=\ \lambda n\,.\,n\,\mathsf{suc}\,\underline{0}\ :\ \tau^{**} \to \tau^*$$

with the property $\downarrow \underline{n} =_\beta \underline{n}$. Note that $\downarrow$ also has the type $\mathsf{Nat_0}^* \to \mathsf{Nat}_0$, since suc can be typed as $\mathsf{Nat}_0 \to \mathsf{Nat}_0$ by instantiating the argument $n : \mathsf{Nat}_0$ as $n : {\alpha_0}^*$. Note moreover that add and mult, by a similar argument, can also have type $\mathsf{Nat}_0 \to \mathsf{Nat}_0 \to \mathsf{Nat}_0$.

The predecessor can be implemented of type $\mathsf{Nat}_0 \to \mathsf{Nat}_0$ as follows: in the context where we have variables $s : \alpha_0 \to \alpha_0$ and $z : \alpha_0$, as abstract successor and zero, we get the term $P := \lambda p\,.\,\langle s(p\mathsf{L}), p\mathsf{L}\rangle$ of type $(\alpha_0 \times \alpha_0) \to (\alpha_0 \times \alpha_0)$, such that the n-fold iteration of P applied to $\langle z, z\rangle$ reduces to $\langle s^n z, s^{n \dot{-} 1} z\rangle$, for every $n \geq 0$. Thus the argument $n : \mathsf{Nat}_0$ is instantiated as $n : (\alpha_0 \times \alpha_0)^*$ by the rule $(\forall E^2)$, and we get

$$n : \mathsf{Nat}_0 \vdash \lambda sz\,.\,nP\langle z, z\rangle\mathsf{R} : {\alpha_0}^*$$

and an application of $(\forall I)$ and $(\to I)$ yields that the predecessor

$$\mathsf{pred} := \lambda nsz\,.\,nP\langle z, z\rangle\mathsf{R}$$

is typeable as $\mathsf{pred} : \mathsf{Nat}_0 \to \mathsf{Nat}_0$.

We obtain subtraction $\mathsf{sub} := \lambda mn\,.\,n\,\mathsf{pred}\,m$ by iterating the predecessor, of type $\mathsf{sub} : \mathsf{Nat}_0 \to \mathsf{Nat_0}^* \to \mathsf{Nat}_0$. Obviously, for $m, n \in \mathbb{N}$ we have $\mathsf{sub}\ \underline{m}\,\underline{n} =_{\beta\eta} \underline{m \dot{-} n}$.

Testing for zero can also be easily programmed as $\chi_0 := \lambda nxy\,.\,n(\lambda z\,.\,y)x$, which has type $\chi_0 : \mathsf{Nat}_0 \to \alpha_0 \to \alpha_0 \to \alpha_0$, and the operational semantics *if* $n = 0$ *then* x *else* y, i.e., with the properties $\chi_0\,\underline{0}\,x\,y =_\beta x$ and $\chi_0\,\underline{n+1}\,x\,y =_\beta y$. To obtain the typing, we instantiate the input $n : \mathsf{Nat}_0$ as $n : {\alpha_0}^*$ by $(\forall E^2)$.

Next we define a function T_0 such that for natural numbers n and m, we have $T_0\,\underline{0}\,\underline{m} =_{\beta\eta} \underline{m}$, and $T_0\,\underline{n+1}\,\underline{m} =_{\beta\eta} \underline{m+1}$, as

$$T_0 := \lambda nxszs'z'\,.\,\chi_0\,n(s(xsz)s'z')(xszs'z')$$

The term T_0 can have the type $\mathsf{Nat}_0 \to \mathsf{Nat_0}^* \to \mathsf{Nat_0}^*$, which is verified as follows: in the context $x : \mathsf{Nat_0}^*$, $s : \mathsf{Nat}_0 \to \mathsf{Nat}_0$, $z : \mathsf{Nat}_0$ we obtain the terms xsz and $s(xsz)$ of type Nat_0. These are instantiated with the rule $(\forall E^2)$ as being of type ${\alpha_0}^*$, and with $s' : \alpha_0 \to \alpha_0$ and $z' : \alpha_0$ we get $s(xsz)s'z' : \alpha_0$ and $xszs'z' : \alpha_0$. Therefore we obtain

$$\Gamma \vdash \lambda s'z'\,.\,\chi_0\,n(s(xsz)s'z')(xszs'z') : {\alpha_0}^*\ ,$$

where Γ is the context $n : \mathsf{Nat}_0, x : \mathsf{Nat}_0{}^*, s : \mathsf{Nat}_0 \to \mathsf{Nat}_0, z : \mathsf{Nat}_0$, and an application of $(\forall I)$ followed by several $(\to I)$ gives the claimed typing of T_0. It is easily verified by straightforward calculations that T_0 has the claimed operational behaviour.

We use T_0 to implement an upward typecast that works with the aid of a large parameter of suitable type, i.e., a term $\uparrow : \mathsf{Nat}_0{}^{**} \to \mathsf{Nat}_0 \to \mathsf{Nat}_0{}^*$ with the property that $\uparrow \underline{m}\,\underline{n} =_{\beta\eta} \underline{n}$ as long as $m \geq n$. This can be implemented as

$$\uparrow := \lambda mn \,.\, m\big(\lambda x \,.\, T_0(\mathsf{sub}\, nx)x\big)\underline{0} \;,$$

i.e., the function $\lambda x \,.\, T_0(\mathsf{sub}\, nx)x$, which operationally behaves as

$$\textit{if } x < n \textit{ then } x + 1 \textit{ else } x,$$

is iterated m times, starting at 0, to the effect that in the first n iterations, the value is increased by 1, and thereafter the value is n, and thus remains the same.

Now by use of the typecast, a more useful type-homogeneous subtraction, but again with the aid of a large parameter, can be defined as

$$\widetilde{\mathsf{sub}} := \lambda mnk \,.\uparrow m(\mathsf{sub}(\downarrow n)k) \;:\; \mathsf{Nat}_0{}^{**} \to \mathsf{Nat}_0{}^* \to \mathsf{Nat}_0{}^* \to \mathsf{Nat}_0{}^* \;,$$

with the property that $\widetilde{\mathsf{sub}}\, \underline{m}\, \underline{n}\, \underline{k} =_{\beta\eta} \underline{n \dot{-} k}$ as long as $m \geq n \dot{-} k$.

Definition 3.1. For a type τ, let $\tau^{(0)} := \tau$, and $\tau^{(k+1)} := \left(\tau^{(k)}\right)^*$.

To iterate the above construction, assume we have a subtraction

$$\widetilde{\mathsf{sub}}_k : \mathsf{Nat}_0{}^{(k+1)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k)} \;,$$

and note that T_0 can have type $\mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k+1)} \to \mathsf{Nat}_0{}^{(k+1)}$ for every k, since χ_0 can have type $\mathsf{Nat}_0{}^{(k+1)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k)}$ (in fact, χ_0 can have any type of the form $\tau^* \to \tau \to \tau \to \tau$). Thus we can program an upward typecast

$$\uparrow_k := \lambda mn \,.\, m\big(\lambda x \,.\, T_0(\widetilde{\mathsf{sub}}_k(\downarrow m)n(\downarrow x))x\big)\underline{0}$$

of type $\uparrow_k : \mathsf{Nat}_0{}^{(k+2)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k+1)}$, which again can be used to define a subtraction

$$\widetilde{\mathsf{sub}}_{k+1} := \lambda m n_1 n_2 \,.\uparrow_k m\big(\widetilde{\mathsf{sub}}_k(\downarrow m)(\downarrow n_1)(\downarrow n_2)\big)$$

of type $\mathsf{Nat}_0{}^{(k+2)} \to \mathsf{Nat}_0{}^{(k+1)} \to \mathsf{Nat}_0{}^{(k+1)} \to \mathsf{Nat}_0{}^{(k+1)}$. Thus inductively we get subtractions $\widetilde{\mathsf{sub}}_k$ and upward typecasts $\uparrow_k$ for every k. We also define iterated upward typecasts $\uparrow_k^\ell : \mathsf{Nat}_0{}^{(k+\ell+1)} \to \mathsf{Nat}_0{}^{(k)} \to \mathsf{Nat}_0{}^{(k+\ell)}$ by

$$\uparrow_k^0 := \lambda mn \,.\, n \qquad \text{and} \qquad \uparrow_k^{\ell+1} := \lambda mn \,.\uparrow_{k+\ell} m(\uparrow_k^\ell(\downarrow m)n) \;.$$

From now on we will omit the index k in $\uparrow_k$, $\uparrow_k^\ell$ and $\widetilde{\mathsf{sub}}_k$ when it can be inferred from the context. We are ready to state our main lemma.

LEMMA 3.2. *For every elementary recursive function $f : \mathbb{N}^n \to \mathbb{N}$ and $k \in \mathbb{N}$, there are a closed term t and $\ell, r \in \mathbb{N}$ and a list $\vec{\eta}$ of types, where each η is of the form $\eta ::= \mathsf{Nat}_0{}^{(k)} \mid \eta^* \mid (\eta \times \eta)^*$, such that*

$$\vdash t : \vec{\eta} \to \overrightarrow{\mathsf{Nat}_0{}^{(k+\ell)}} \to \mathsf{Nat}_0{}^{(k)}$$

and for all $\vec{n} \in \mathbb{N}^n$, $t\, \vec{\underline{L}}\, \vec{\underline{n}} \;=_{\beta\eta}\; \underline{f(\vec{n})}$ as long as $L \geq 2_r(\sum \vec{n})$.

Note that we plug in the same numeral $\underline{L}$ for all the arguments of the types $\vec{\eta}$. Also note that only simple types over Nat_0 are used as these types $\vec{\eta}$, and this is the only property used in the application and proof. A statement similar to this lemma was offered by Simmons [2004] as a characterization of the Kalmár elementary recursive functions.

Before we prove the main lemma, we shall first use it to derive the main theorem of this section, the representability of all elementary recursive functions.

THEOREM 3.3. *For every elementary recursive function* $f\colon \mathbb{N}^n \to \mathbb{N}$ *there is a closed term* $T : \mathsf{Nat}_1^n \to \mathsf{Nat}_0$ *such that* $T\,\underline{\vec{n}} =_{\beta\eta} \underline{f(\vec{n})}$ *for all* $\vec{n} \in \mathbb{N}^n$*.*

PROOF. From the lemma for f and $k = 1$, we obtain a term t and $\ell, r, \vec{\eta}$ with the properties stated there. As we can always move to bigger values of r, we may without loss of generality assume r to be at least 2 and even.

Let $s := r/2$. For each type η_i, instantiate each input $n : \mathsf{Nat}_1$ as $n : {\eta_i}^{(s)}$, which is possible by $(\forall E^1)$ since ${\eta_i}^{(s-1)}$ is a closed type of level 1. Now use $\mathsf{add} : {\eta_i}^{(s)} \to {\eta_i}^{(s)} \to {\eta_i}^{(s)}$ to compute $S := \sum \vec{n}$ of type ${\eta_i}^{(s)}$. Next form the term $N := (\ldots((S\underline{2})\underline{2})\ldots\underline{2})$, with r occurrences of the numeral $\underline{2}$, of type η_i.

Instantiate the inputs $\vec{n}$ again by $(\forall E^1)$ at the closed, level 1 type ${\mathsf{Nat}_0}^{(\ell)}$, and form $T := \lambda\vec{n}\,.\downarrow\!(t\,\vec{N}\,\vec{n})$. As for every input $\vec{n}$, N evaluates to a numeral $\underline{L}$ with $L \geq 2_r(\sum\vec{n})$, the term T has the required properties, by the lemma. $\square$

COROLLARY 3.4. *For every elementary recursive function* $f\colon \mathbb{N}^n \to \mathbb{N}$ *there is a closed term* $T : \mathsf{Nat}_1^n \to \mathsf{Nat}_0$ *such that* $T\,\underline{\vec{n}} \leadsto \underline{f(\vec{n})}$ *for all* $\vec{n} \in \mathbb{N}^n$*.*

PROOF. Take $\lambda\vec{n}sz\,.\,T\vec{n}sz$ for the term T obtained from the theorem. For every $\vec{n}$ consider the β-normal of $(\lambda\vec{n}sz\,.\,T\vec{n}sz)\underline{\vec{n}}$, which is a closed, β-normal term of type $\mathbb{N}$, starting with two abstractions, hence a numeral. So it has to be $\underline{f(\vec{n})}$, for otherwise two distinct numeral would be $\beta\eta$-equal, which by the well known confluence of lambda calculus with pairs is not the case. $\square$

PROOF OF THE LEMMA. We have produced terms representing the base functions successor, addition, subtraction and multiplication above. For S, $+$ and $\times$, we can set $\ell = r = 0$ and $\vec{\eta}$ empty for any k.

Concerning subtraction $\dot{-}$, for $k = 0$ we use the term $\lambda nk\,.\,\mathsf{sub}(\downarrow n)k$ and set $r = 0$, $\ell = 1$ and $\vec{\eta}$ empty, and for $k \geq 1$ we use $\widetilde{\mathsf{sub}}_k$, and we set $\ell = r = 0$ and $\vec{\eta}$ contains the single type ${\mathsf{Nat}_0}^{(k+1)}$.

In the following, note that by the properties of $\downarrow$, whenever we have a term t of type ${\mathsf{Nat}_0}^{(k+\ell)}$, we can obtain a term $\downarrow^{\ell} t$ with the same value of type ${\mathsf{Nat}_0}^{(k)}$.

For closure under composition, let $f(\vec{n}) = g(\overrightarrow{h_{\cdot}(\vec{n})}\,)$ and k be given. By the induction hypothesis for g and k, we have a term t_g, numbers ℓ_g and r_g and a list $\overrightarrow{\eta}$ of types such that the claim of the lemma holds for these.

Also, the induction hypothesis for each h_i and $k+\ell_g$ yields terms t_i and $\ell_i, r_i \in \mathbb{N}$ and types $\overrightarrow{\eta_{\cdot,i}}$, such that the claim holds for these.

Let $\ell := \ell_g + \max_i \ell_i$. Since the functions h_i are elementary recursive, $\sum h_i(\vec{n})$ is also elementary, and therefore there is an $s \in \mathbb{N}$ such that $\sum_i h_i(\vec{n}) \leq 2_s(\sum\vec{n})$. For variables $\vec{v}$ and $\overrightarrow{w_{\cdot,i}}$, which we give the types $\vec{\eta}$ and $\overrightarrow{\eta_{\cdot,i}}$, respectively, we set

$$t := \lambda\vec{v}\;\vec{\vec{w}}\;\vec{n}\;.\;t_g\;\vec{v}\;\overrightarrow{(t_{\cdot}\;\overrightarrow{w_{\cdot,\cdot\cdot}}\;\overrightarrow{(\downarrow^{\ell-\ell_g-\ell_{\cdot\cdot}}\;n_{\cdot})}\;)}$$

such that t has type $\vec{\eta} \to \vec{\vec{\eta}} \to \overrightarrow{\mathsf{Nat}_0{}^{(k+\ell)}} \to \mathsf{Nat}_0{}^{(k)}$. By the induction hypothesis, for $r := \max(r_g + s, \vec{r})$ we have $t\,\underline{\vec{L}}\,\underline{\vec{\vec{L}}}\,\underline{\vec{n}} =_{\beta\eta} \underline{f(\vec{n})}$ as long as $L \geq 2_r(\sum \vec{n})$.

For closure under bounded sums, let $f(\vec{n}, m) = \sum_{i=0}^{m-1} g(\vec{n}, i)$ and k be given. By the induction hypothesis for g and $k+1$, we have a term t_g, numbers ℓ and r and a list $\vec{\eta}$ of types such that the claim of the lemma holds. Define

$$\tilde{\chi}_0 := \lambda nxysz\,.\,\chi_0\, n\,(xsz)\,(ysz)$$

of type $(\mathsf{Nat}_0{}^{(k+\ell+1)})^3 \to \mathsf{Nat}_0{}^{(k+\ell+1)}$, with the same operational semantics as χ_0, i.e., for $i, j \in \mathbb{N}$ we have $\tilde{\chi}_0\,\underline{0}\,\underline{i}\,\underline{j} =_{\beta\eta} \underline{i}$ and $\tilde{\chi}_0\,\underline{n+1}\,\underline{i}\,\underline{j} =_{\beta\eta} \underline{j}$. For variables $v : \mathsf{Nat}_0{}^{(k+\ell+2)}$, $\vec{w}$ of the types $\vec{\eta}$ and $\vec{n}, m$ of type $\mathsf{Nat}_0{}^{(k+\ell+1)}$, we have

$$T := \lambda xy\,.\,\tilde{\chi}_0\,(\widetilde{\mathsf{sub}}\,v\,m\,y)x(\mathsf{add}\,x(\uparrow^\ell\, v\,(t_g\,\vec{w}\,\vec{n}\,y)))\ ,$$

of type $\mathsf{Nat}_0{}^{(k+\ell+1)} \to \mathsf{Nat}_0{}^{(k+\ell+1)} \to \mathsf{Nat}_0{}^{(k+\ell+1)}$.

As long as a sufficiently large numeral $\underline{L}$ is substituted for the variables v and $\vec{w}$, T operationally behaves as

if $y < m$ *then* $x + g(\vec{n}, y)$ *else* x.

More precisely, L has to be large enough so that all values of $g(\vec{n}, i)$ are computed correctly, that is, $L \geq 2_r(\sum \vec{n} + m)$, and we need $L \geq g(\vec{n}, i)$ for the typecast $\uparrow^\ell$ to work properly. Next, we define

$$P := \lambda p\,.\,\langle T(p\mathsf{L})\,(p\mathsf{R}), \mathsf{suc}(p\mathsf{R})\rangle$$

of type $(\mathsf{Nat}_0{}^{(k+\ell+1)} \times \mathsf{Nat}_0{}^{(k+\ell+1)}) \to (\mathsf{Nat}_0{}^{(k+\ell+1)} \times \mathsf{Nat}_0{}^{(k+\ell+1)})$. When this term, having the operational semantics

$$\langle s, i\rangle \mapsto \begin{cases} \langle s + g(\vec{n}, i), i+1\rangle & \text{if } i < m \\ \langle s, i+1\rangle & \text{otherwise,} \end{cases}$$

is iterated starting from the pair $\langle 0, 0\rangle$, by use of a sufficiently large numeral of type $(\mathsf{Nat}_0{}^{(k+\ell+1)} \times \mathsf{Nat}_0{}^{(k+\ell+1)})^*$, the values $g(\vec{n}, i)$ for $i = 0, \ldots, m-1$ are summed up in the left component. Thus to represent f, we define the term

$$t := \lambda u\,v\,\vec{w}\,\vec{n}\,m\,.\,{\downarrow^{\ell+1}}\big(u\,P\,\langle\underline{0},\underline{0}\rangle\mathsf{L}\big)$$

of type

$$(\mathsf{Nat}_0{}^{(k+\ell+1)} \times \mathsf{Nat}_0{}^{(k+\ell+1)})^* \to \mathsf{Nat}_0{}^{(k+\ell+2)} \to \vec{\eta} \to \overrightarrow{\mathsf{Nat}^{(k+\ell+1)}} \to \mathsf{Nat}_0{}^{(k)}\ .$$

By the induction hypothesis and the construction, we get the property that $t\,\underline{L}\,\underline{L}\,\underline{\vec{L}}\,\underline{\vec{n}}\,\underline{m} \;=_{\beta\eta} \underline{f(\vec{n}, m)}$ as long as L is sufficiently large. To be more precise, L needs to satisfy the requirements above for T to be computed correctly, and $L \geq m$ in order to complete the summation. Therefore, let s be such that for every m and $i \leq m$ we have $g(\vec{n}, i) \leq 2_s(\sum \vec{n} + m)$, which exists since g is elementary recursive, and let $r' := \max(r, s)$. Then all conditions on L are satisfied if $L \geq 2_{r'}(\sum \vec{n} + m)$.

Closure under bounded products is shown in the same way, only with add in the definition of T replaced by mult, and the iteration of P is started at $\langle 1, 0\rangle$. $\square$

4. SOUNDNESS

In this section we show the other direction of our claim, that is, we show that every term of type $\mathsf{Nat}_1 \to \mathsf{Nat}_0$ denotes a function on Church numerals computable in elementary space. The main idea is to use the elementary bound for traditional cut-elimination in propositional logic. In this section we will deal only with types of level at most 1, so let τ, ρ, σ range over those types within this section. Note that every instantiation of Nat_1 is a type of level 1. Types of level 0 and 1 are almost simple types (corresponding to propositional logic) with the exception of quantification of α_0. These quantifiers however, can only be instantiated with flat types of the form $\alpha_0 \times \ldots \times \alpha_0$. Hence we can get a notion of cut-rank that is invariant under generalization and instantiation of level 0, if we ignore pairs. Fortunately we can do so, as the reduction of a pair-redex reduces the size of the term and hence does not do any harm. So we define the rank $\mathrm{rk}(\tau)$ of a type τ inductively as follows:

$$\begin{aligned}
\mathrm{rk}(\alpha) &:= 0 \\
\mathrm{rk}(\rho \times \sigma) &:= \max(\mathrm{rk}(\rho), \mathrm{rk}(\sigma)) \\
\mathrm{rk}(\rho \to \sigma) &:= \max(\mathrm{rk}(\rho) + 1, \mathrm{rk}(\sigma)) \\
\mathrm{rk}(\forall \alpha . \rho) &:= \mathrm{rk}(\rho)
\end{aligned}$$

We inductively define a relation $\Gamma \vdash^m_k r : \tau$ saying that $\Gamma \vdash r : \tau$ can be derived by a typing derivation of height m and cut-rank k.

$$(\mathrm{var})\ \frac{}{\Gamma \vdash^m_k x : \tau} \qquad \text{if } x : \tau \text{ occurs in } \Gamma \text{ and } m, k \geq 0$$

$$(\to I)\ \frac{\Gamma, x : \sigma \vdash^m_k r : \rho}{\Gamma \vdash^{m+1}_k \lambda x \,.\, r : \sigma \to \rho}$$

$$(\to E)\ \frac{\Gamma \vdash^m_k r : \sigma \to \rho \qquad \Gamma \vdash^{m'}_k s : \sigma}{\Gamma \vdash^{m''}_k rs : \rho} \qquad \text{if } \mathrm{rk}(\sigma) < k$$

$$(\times I)\ \frac{\Gamma \vdash^m_k r : \rho \qquad \Gamma \vdash^{m'}_k s : \sigma}{\Gamma \vdash^{m''}_k \langle r, s \rangle : \rho \times \sigma}$$

$$(\times E^1)\ \frac{\Gamma \vdash^m_k r : \sigma \times \rho}{\Gamma \vdash^{m+1}_k r\mathsf{L} : \sigma} \qquad \text{and analogous for } (\times E^2)$$

$$(\forall I)\ \frac{\Gamma \vdash^m_k r : \tau}{\Gamma \vdash^{m+1}_k r : \forall \alpha . \tau} \qquad \text{if } \alpha \text{ does not occur free in} \Gamma$$

$$(\forall E^2)\ \frac{\Gamma \vdash^m_k r : \forall \alpha . \tau}{\Gamma \vdash^{m+1}_k r : \tau[\alpha := \sigma']} \qquad \text{where } \sigma' \text{ is a flat type.}$$

where $m'' := \max(m, m') + 1$. As the rules are precisely those of our typing judgment for types of level at most 1, we have the following property for typing derivations of level at most 1: if $\Gamma \vdash r : \tau$ then there are m, k such that $\Gamma \vdash^m_k r : \tau$. On the other hand, the following property obviously holds and motivates our interest in this notion.

Proposition 4.1. *If* $\Gamma \vdash^m_k r : \tau$, *then* $|r| \leq 2^m$.

The rules are formulated in such a way that weakening is admissible.

Proposition 4.2 (Weakening). *If* $\Gamma \vdash^m_k r : \tau$, $\Gamma' \supset \Gamma$, $m' \geq m$, $k' \geq k$ *then* $\Gamma' \vdash^{m'}_{k'} r : \tau$.

The next proposition, which can be shown by a trivial induction on τ'_0 or τ, respectively, explains formally why we can allow instantiations with flat types of level 0 without any harm: the rank is not altered!

Proposition 4.3. *For a flat type* τ'_0 *of level* 0 *we have* $\mathrm{rk}(\tau'_0) = 0$ *and* $\mathrm{rk}(\tau[\alpha_0 := \tau'_0]) = \mathrm{rk}(\tau)$.

Knowing that the rank of a type is not altered by substituting in a flat type, the cut-rank, being a rank, is not altered as well, hence an induction on $\Gamma \vdash^m_k t : \tau$ shows the following.

Proposition 4.4. *If* $\Gamma \vdash^m_k t{:}\tau$ *and* τ'_0 *is a flat type of level* 0 *then* $\Gamma[\alpha_0 := \tau'_0] \vdash^m_k t : \tau[\alpha_0 := \tau'_0]$

Using this proposition a simple induction on m shows that a derivation $\Gamma \vdash^m_k t : \tau$ can be transformed in such a way that the rule $(\forall I)$ is never followed by $(\forall E^2)$. So from now on we tacitly assume all derivations to be free from those $(\forall I)$-$(\forall E^2)$-redexes, as for example in the proof of the next proposition, which then is a simple analysis of the last rule of the derivation.

Proposition 4.5. *If* $\Gamma \vdash^m_k \langle r, s\rangle \mathsf{L} : \rho$ *then* $\Gamma \vdash^m_k r : \rho$ *and if* $\Gamma \vdash^m_k \langle r, s\rangle \mathsf{R} : \sigma$ *then* $\Gamma \vdash^m_k s : \sigma$.

As usual, induction on the first derivation shows that cuts can be performed at the cost of summing up heights.

Lemma 4.6. *If* $\Gamma, x : \rho \vdash^m_k s : \sigma$ *and* $\Gamma \vdash^{m'}_k r : \rho$ *then* $\Gamma \vdash^{m+m'}_k s[x := r] : \sigma$.

In order to be able to reduce the cut rank, we first show an "inversion"-lemma, that is, we show that under certain conditions terms of arrow-type can be brought into abstraction form.

Lemma 4.7 (Inversion). *If* $\mathrm{rk}(\Gamma) \leq k$ *and* $\Gamma \vdash^m_k t : \rho \to \sigma$ *where* $\mathrm{rk}(\rho) \geq k$, *then there are* t' *and* x *with* $t =_\beta \lambda x\,.\,t'$ *such that* $\Gamma, x : \rho \vdash^m_k t' : \sigma$.

Proof. Induction on m and case distinction according to t.

The case $t = x\vec{s}$ is impossible, since x would have to occur in Γ and hence $\mathrm{rk}(\Gamma) > k$. The case $t = \langle r, s\rangle\vec{t}$ is also impossible since $\vec{t}$ has to be empty, as we assume t to be free of pair-redexes, and therefore t would have to have a pair type.

So the only remaining case is that t is of the form $t = (\lambda y\,.\,r)\vec{t}$. The claim is trivial if $\vec{t}$ is empty. So without loss of generality we might assume t to be $t = (\lambda y\,.\,r)s\vec{s}$, with y not free in $s, \vec{s}$. The abstraction $\lambda y\,.\,r$ must have been introduced from a derivation $\Gamma, y:\tau \vdash_k^m r:\tilde{\tau}$ with $\mathrm{rk}(\tau) < k$ for otherwise the cut would not have been allowed. Hence, for some m' with $m'+2 \leq m$ we get

$$\Gamma, y:\tau \vdash_k^{m'} r\vec{s}:\rho \to \sigma \text{ and } \Gamma \vdash_k^{m'+1} s:\tau$$

Hence by the induction hypothesis we get a new variable x and a term t' such that $r\vec{s} =_\beta \lambda x\,.\,t'$ and $\Gamma, y:\tau, x:\rho \vdash_k^{m'} t':\sigma$. From that we conclude $\Gamma, x:\rho \vdash_k^{m'+2} (\lambda y\,.\,t')s:\sigma$ and note $\lambda x\,.\,(\lambda y\,.\,t')s =_\beta \lambda x\,.\,t'[y:=s] = (\lambda x\,.\,t')[y:=s] =_\beta (r\vec{s})[y:=s] = r[y:=s]\vec{s} =_\beta (\lambda y\,.\,r)s\vec{s} = t$, hence the claim. $\square$

Lemma 4.8 (Cut-rank reduction). *If $\Gamma \vdash_{k+1}^m t:\rho$, $\mathrm{rk}(\Gamma) \leq k$, and $\mathrm{rk}(\rho) \leq k+1$ then $\Gamma \vdash_k^{2^m} t':\rho$ for some $t' =_\beta t$.*

Proof. Induction on m. The only interesting cases are $(\to I)$ and $(\to E)$. Concerning $(\to I)$ we are in the situation that $\Gamma \vdash_{k+1}^{m+1} \lambda x\,.\,r:\sigma\to\tau$ was concluded from $\Gamma, x:\sigma \vdash_{k+1}^m r:\tau$. With $\rho = \sigma\to\tau$ we have $\mathrm{rk}(\Gamma) \leq k$, $\mathrm{rk}(\sigma) < \mathrm{rk}(\rho) \leq k+1$ and $\mathrm{rk}(\tau) \leq \mathrm{rk}(\rho) \leq k+1$. Hence an application of the induction hypothesis yields $\Gamma, x:\sigma \vdash_k^{2^m} r':\tau$ from which we conclude $\Gamma \vdash_k^{2^m+1} \lambda x\,.\,r':\sigma\to\tau$ which, by weakening, suffices, since $2^m+1 \leq 2^{m+1}$.

Concerning the case $(\to E)$ we are in the situation that $\Gamma \vdash_{k+1}^{m+1} ts{:}\rho$ was concluded from $\Gamma \vdash_{k+1}^m t:\sigma\to\rho$ and $\Gamma \vdash_{k+1}^m s:\sigma$. The only case that is not immediate by the induction hypothesis is if $\mathrm{rk}(\sigma) = k$. Then the induction hypothesis gives us $\Gamma \vdash_k^{2^m} t':\sigma\to\rho$ for some $t' =_\beta t$. By our assumption $\mathrm{rk}(\Gamma) \leq k$, hence by inversion we get $\Gamma, x:\sigma \vdash_k^{2^m} t'':\rho$ for some new x and t'' such that $\lambda x\,.\,t'' =_\beta t' =_\beta t$. Also by the induction hypothesis we get $\Gamma \vdash_k^{2^m} s:\sigma$. By Lemma 4.6 we get $\Gamma \vdash_k^{2^m+2^m} t''[x:=s]:\rho$ which yields the claim since $t''[x:=s] =_\beta (\lambda x\,.\,t'')s =_\beta ts$. $\square$

Corollary 4.9. *If $\vdash_{k+1}^m t:\alpha^*$ then $\vdash_1^{2_k(m)} t':\alpha^*$ for some $t' =_\beta t$.*

Proposition 4.10. *If t normal and $\Gamma \vdash t:\tau_0'$ for some Γ with $\mathrm{rk}(\Gamma) \leq 1$ then t is λ-free.*

Proof. Inspection of the typing rules yields that the only rule introducing a λ is $(\to I)$, which creates an arrow-type. In order for the whole term to be of arrow-free type, the rule $(\to E)$ has to be used, either creating a redex or requiring a variable of rank at least 2. $\square$

Definition 4.11. A term t is *quasinormal*, if every redex in t is of the form $\langle r,s\rangle\mathsf{L}$ or $\langle r,s\rangle\mathsf{R}$ with λ-free r and s.

We remark the trivial property that the normal form of a quasinormal term t can be computed in space bound by the length of t. We also note that Proposition 4.10 also holds for quasinormal terms, since the only types discarded by a redex are those of terms which are λ-free by definition. Moreover, a simple induction on t shows

Proposition 4.12. *If t is quasinormal and s is λ-free and quasinormal then $t[x:=s]$ is quasinormal.*

From that proposition, Proposition 4.10 and Lemma 4.6 the following is immediately obtained.

Corollary 4.13. *If* $\Gamma, x : \sigma' \vdash_1^m r{:}\rho$ *and* $\Gamma \vdash_1^{m'} s{:}\sigma'$ *and* r *and* s *are quasinormal then* $\Gamma \vdash_1^{m+m'} r[x := s] : \rho$ *and* $r[x := s]$ *is quasinormal.*

This corollary allows us to show our last ingredient for the soundness theorem: we can transform a term with cut-rank 1 into a quasinormal one at exponential cost.

Lemma 4.14. *If* $\Gamma \vdash_1^m t{:}\tau$ *then* $\Gamma \vdash_1^{2^m} t'{:}\tau$ *for some quasinormal* t' *with* $t' =_\beta t$.

Proof. Induction on m. If t is not quasinormal, it has a subterm of the form $(\lambda x\,.\,r)s$. Then, for some Δ, σ, ρ and k we have $\Delta, x : \sigma \vdash_1^k r : \rho$, and $\Delta \vdash_1^{k+1} s : \sigma$ from which $\Delta \vdash_1^{k+2} (\lambda x\,.\,r)s : \rho$ was concluded. Since the cut was allowed, we have $\mathrm{rk}(\sigma) < 1$. Hence, by the induction hypotheses we get a quasinormal $s' =_\beta s$ such that $\Delta \vdash_1^{2^{k+1}} s' : \sigma$. Also by induction hypothesis we get a quasinormal $r' =_\beta r$ such that $\Delta, x : \sigma \vdash_1^{2^k} r' : \rho$. By Corollary 4.13 we get $\Delta \vdash_1^{2^k+2^{k+1}} r'[x := s'] : \rho$ and $r'[x := s']$ is quasinormal, hence the claim. □

We are now ready to show that every representable function is elementary recursive. To keep the notation simple, we only state and prove this for unary functions, but the generalization to higher arities is straightforward.

Theorem 4.15. *If* $\vdash t : \mathsf{Nat}_1 \to \mathsf{Nat}_0$ *then* t *denotes an elementary function on Church numerals.*

Proof. We have $x : \mathsf{Nat}_1 \vdash tx : \mathsf{Nat}_0$. Since all our terms are also typeable in usual system F, hence strongly normalizing, and since subject reduction holds, we can find (in maybe long time, which however is independent of the input) a normal term $t' =_\beta tx$ and $x : \mathsf{Nat}_1 \vdash t' : \mathsf{Nat}_0$. Since t' is normal, inspection of the typing rules yields that every occurrence of x must be within some context, that is, of the form

$$(\forall E^1) \quad \frac{x : \mathsf{Nat}_1 \vdash x : \mathsf{Nat}_1}{x : \mathsf{Nat}_1 \vdash x : \xi^*}$$

for some level 1 type ξ, without (free) variable α_1. Let c be the maximum of the ranks of all the ξ's occurring in that derivation and k the number of occurrences of such ξ's (note that c and k are still independent of the input).

Now, let a natural number n be given. Replacing all $x{:}\xi^*$ by derivations of $\underline{n}{:}\xi^*$ yields a term $t'' =_\beta t\underline{n}$ and a derivation $\vdash_c^{k\cdot(n+2)+2|t'|} t'' : \mathsf{Nat}_0$. The bound on the height of the derivation is obtained as follows: there are k derivations of height $n{+}2$ yielding $n : \xi^*$ and these are plugged into the derivation of $t' : \mathsf{Nat}_0$. In the latter derivation there is at most one inference for each symbol in t' followed possible by a single quantifier inference.

Using Corollary 4.9 we obtain a term $\tilde{t} =_\beta t'' =_\beta t\underline{n}$ such that $\vdash_1^{2_{c+1}(k(n+2)+2|t'|)} \tilde{t} : \mathsf{Nat}_0$. Hence Lemma 4.14 and the remark on computing the normal form of a quasinormal term provides means to calculate the normal form of $t\underline{n}$ in elementary space. (Note that all the intermediate terms are also of elementary bounded size.) □

Together with Theorem 3.3 we obtain the claimed characterization.

Corollary 4.16. *The representable functions are precisely the elementary recursive functions.*

Note that our characterization does not mean that the normalization procedure for terms typeable in our system is elementary recursive. The following easy counterexample shows that this is indeed not the case: the terms $(\ldots((\underline{2}\,\underline{2})\underline{2})\ldots\underline{2})$ with n occurrences of $\underline{2}$ are of size $O(n)$, but their normal forms are the numerals $\underline{2_n(1)}$ of size $\Omega(2_n(1))$. Thus the normalization function has super-elementary growth.

REFERENCES

ALTENKIRCH, T. AND COQUAND, T. 2001. A finitary subsystem of the polymorphic lambda-calculus. In *Typed Lambda Calculi and Applications*, S. Abramsky, Ed. LNCS, vol. 2044. Springer, 22–28.

ASPERTI, A. AND ROVERSI, L. 2002. Intuitionistic light affine logic. *ACM Transactions on Computational Logic 3,* 1, 137–175.

CLOTE, P. 1999. Computation models and function algebras. In *Handbook of Computability Theory*, E. R. Griffor, Ed. Elsevier, 589–681.

DANOS, V. AND JOINET, J.-B. 2003. Linear logic and elementary time. *Information and Computation 183,* 1, 123–137.

GIRARD, J.-Y. 1971. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In *Proceedings of the 2nd Scandinavian Logic Symposium*, J. Fenstad, Ed. North-Holland, 63–92.

GIRARD, J.-Y. 1998. Light linear logic. *Information and Computation 143,* 2, 175–204.

GRZEGORCZYK, A. 1953. Some classes of recursive fuctions. *Rozprawy Matematyczne 4.*

KALMÁR, L. 1943. Egyszerű példa eldönthetetlen aritmetikai problémára. *Matematikai és Fizikai Lapok 50*, 1–23.

LEIVANT, D. 1991. Finitely stratified polymorphism. *Information and Computation 93*, 93–113.

REYNOLDS, J. 1974. Towards a theory of type structure. In *Proceedings, Colloque sur la programmation.* Springer LNCS 19, 408–425.

SIMMONS, H. 2004? Tiering as a recursion technique. *Bulletin of Symbolic Logic*, to appear.

Received October 2002; revised November 2003 and March 2004; accepted April 2004