Higher Types, Finite Domains and Resource-bounded Turing Machines

LARS KRISTIANSEN, *Department of Mathematics, University of Oslo, PO Box 1053, Blindern, NO-0316 Oslo, Norway.*
E-mail: larsk@math.uio.no

Abstract

We prove that neat and natural fragments of the higher order programming language, PCF, capture complexity classes defined by imposing resource bounds on Turing machines. Moreover, we survey some related research on on Gödel's T, and discuss the relationship between fragments of Gödel's T and fragments of PCF. Our proofs are based on denotational semantics and domain theory.

Keywords: Recursion in higher types, complexity theory, domain theory.

1 Introduction and overview

We will attempt to bridge some of the gap between recursion in higher types and domain theory, on the one hand, and complexity theory and resource-bounded Turing machines, on the other hand. The aim of this first section is to motivate the technical work in Sections 2 and 3. We will define some basic notions, survey previous research and discuss the relevance of this research.

1.1 The programming language T^- *and the hierarchy* $\mathcal{G}$

The programming language T^- is based on Gödel's system T. We will assume that the reader is familiar with the typed λ-calculus and Gödel's T. For more on Gödel's T, see [2].

DEFINITION 1
We define the *types* recursively:

- ι is a type (primitive type);
- $\sigma \otimes \tau$ is a type if σ and τ are types (product types);
- $\sigma \to \tau$ is a type if σ and τ are types (arrow types).

The notation $\sigma_1, \sigma_2, \ldots, \sigma_n \to \tau$ is shorthand for $\sigma_1 \to (\sigma_2 \to (\ldots(\sigma_n \to \tau)\ldots))$, and TYP denotes the set of all types.

We define the *terms* of *the typed λ-calculus*:

- We have an infinite supply of variables $x_0^\sigma, x_1^\sigma, x_2^\sigma, \ldots$ for each type σ. A variable of type σ is a term of type σ
- $\lambda x M$ is a term of type $\sigma \to \tau$, if x is a variable of type σ and M is a term of type τ (*λ-abstraction*)
- (MN) is a term of type τ, if M is a term of type $\sigma \to \tau$ and N is a term of type σ (*application*)
- $\langle M, N\rangle$ is a term of type $\sigma \otimes \tau$ if M is a term of type σ and N is a term of type τ (*pairing*)
- $\mathbf{fst}M$ ($\mathbf{snd}M$) is a term of type σ (τ) if M is a term of type $\sigma \otimes \tau$ (*projections*).

Vol. 22 No. 2,
Published online February 17, 2010 doi:10.1093/logcom/exq009

Next we define the *reduction rules* of the typed λ-calculus. We have the following β-conversions: $(\lambda xM)N \rhd M[x:=N]$ if $x \notin FV(N)$; $\mathbf{fst}\langle M,N\rangle \rhd M$; and $\mathbf{snd}\langle M,N\rangle \rhd N$. Further, we have α-conversion, η-conversion and all the other standard reduction rules, i.e. $(MN)\rhd(MN')$ if $N\rhd N'$; $(MN)\rhd(M'N)$ if $M\rhd M'$; ...etc.

The calculus T^- is the typed λ-calculus extended with

- for each $i\in\mathbb{N}$ a constant k_i of type ι
- *recursor terms* $\mathrm{R}_\sigma(G^\sigma, F^{\iota,\sigma\to\sigma}, N^\iota)$ of type σ, i.e. $\mathrm{R}_\sigma(G,F,N)$ is a term of type σ if G and F and N are terms of, respectively, type σ, type $\iota,\sigma\to\sigma$ and type ι
- reduction rules of the form

$$\mathrm{R}_\sigma(G,F,k_0)\rhd G \ \text{ and } \ \mathrm{R}_\sigma(G,F,k_{n+1})\rhd F(k_n,\mathrm{R}_\sigma(G,F,k_n))\,.$$

We will use the standard conventions in the literature: $M[x:=N]$ denotes the term M where every occurrence of the variable x is replaced by the term N; the notations M^σ and $M:\sigma$ are two alternative ways to signify that the term M is of type σ; $M(M_1,M_2)$ denotes the term $((MM_1)M_2)$, etc. If we write M^k, where M is a term and k is a natural number, then M^k denotes the term M repeated k times in a row, e.g. M^3N denotes the term $M(M(M(N)))$. We will use $\overset{\star}{\rhd}$ to denote the transitive–reflexive closure of the reducibility relation $\rhd$; and $=$ to denote the symmetric-transitive–reflexive closure of $\rhd$; and $\equiv$ to denote syntactical equality between terms. ■

As the Definition 1 shows, T^- is more or less a programming language version of Gödels's T where the successor function $S:\iota\to\iota$ is absent. There are various definitions of T^- in the literature. The reason for this is mathematical convenience, and all the definitions are essentially equivalent to the Definition 1.

The programming language T is T^- extended with the successor function $S:\iota\to\iota$ and reduction rules of the form $S(k_i)\rhd k_{i+1}$. It is well known that any closed T-term of type ι, and a fortiori any T^--term of type ι, normalizes to a unique constant k_i. Thus, a closed term $M:\iota\to\iota$ defines a function $f:\mathbb{N}\to\mathbb{N}$, and the value $f(n)$ can be computed by normalizing the term $M(k_n)$. Any function provably total in Peano Arithmetic is definable in T. When we remove the successor function from T, the class of definable functions is of course severely restricted. Indeed, at a first glance it is hard to believe that any interesting functions at all can be defined without the successor function. However, it turns out that T^- is surprisingly powerful when it comes to decision problems. In Definition 2, we stratify the problems decidable in T^- into a hierarchy.

DEFINITION 2
A *problem* is a subset of $\mathbb{N}$. A term $M:\iota\to\iota$ *decides* a problem A when

$$M(k_x)\overset{\star}{\rhd}\begin{cases} k_0 & \text{if } x\in A\\ k_1 & \text{otherwise}\end{cases}$$

We define the *degree of the type* σ, written $\mathrm{dg}(\sigma)$, by recursion on the structure of the type σ:

- $\mathrm{dg}(\iota)=0$
- $\mathrm{dg}(\rho\otimes\tau)=\max(\mathrm{dg}(\rho),\mathrm{dg}(\tau))$
- $\mathrm{dg}(\rho\to\tau)=\max(\mathrm{dg}(\rho)+1,\mathrm{dg}(\tau))$.

The *rank* of a term M, written $\mathrm{Rk}(M)$, is the greatest number n such that $n\le\mathrm{dg}(\sigma)$ for any recursor term $\mathrm{R}_\sigma(\ldots)$ occurring in M.

We define the hierarchy $\mathcal{G}=\bigcup_{n\in\mathbb{N}}\mathcal{G}_n$ by $A\in\mathcal{G}_n$ iff A is decided by a T^--term of rank n. ■

Various schemes have be introduced in order to restrict the power of recursion in higher types. So-called ramification techniques restrict recursion in higher types to the Kalmar elementary level. See e.g. [4, 30]. By using so-called linearity constraints in addition to ramification techniques, such recursion can be restricted further down to the 'polytime' level. The reader should note that we tame the power of recursion in higher types by qualitatively different methods: we remove successor-like functions from a standard computability-theoretic framework.

1.2 T^- *and complexity classes*

We will assume that the reader is familiar with Turing machines and basic complexity theory. For more on these subjects, see [25].

Definition 3

We will work with one-way 1-tape deterministic Turing machines. A Turing machine M *decides* a problem A when M on input $x \in \mathbb{N}$ halts in a distinguished accept state if $x \in A$, and in a distinguished reject state if $x \notin A$. The input $x \in \mathbb{N}$ should be represented in binary on the Turing machine's input tape. Let $|x|$ denote the length of the standard binary representation of the natural number x, and let $2_0^x = x$ and $2_{i+1}^x = 2^{2_i^x}$. For $i \in \mathbb{N}$, we define $\text{TIME } 2_i^{\text{LIN}}$ ($\text{SPACE } 2_i^{\text{LIN}}$) to be the set of problems decidable by a deterministic Turing machine working in TIME (SPACE) $2_i^{k|x|}$ for some fixed $k \in \mathbb{N}$.

It is trivial that $\text{TIME } 2_i^{\text{LIN}} \subseteq \text{SPACE } 2_i^{\text{LIN}}$ and $\text{SPACE } 2_i^{\text{LIN}} \subseteq \text{TIME } 2_{i+1}^{\text{LIN}}$, and thus, we have an *alternating space–time hierarchy*

$$\text{SPACE } 2_0^{\text{LIN}} \subseteq \text{TIME } 2_1^{\text{LIN}} \subseteq \text{SPACE } 2_1^{\text{LIN}} \subseteq \text{TIME } 2_2^{\text{LIN}} \subseteq \text{SPACE } 2_2^{\text{LIN}} \subseteq \text{TIME } 2_3^{\text{LIN}} \subseteq \dots .$$

The three classes at the bottom of the hierarchy are often called, respectively, LINSPACE, EXP and EXPSPACE in the literature. It is well known, and quite obvious, that we have $\text{SPACE } 2_i^{\text{LIN}} \subset \text{SPACE } 2_{i+1}^{\text{LIN}}$ and $\text{TIME } 2_i^{\text{LIN}} \subset \text{TIME } 2_{i+1}^{\text{LIN}}$ for any $i \in \mathbb{N}$. Thus, we know that at least one of the two inclusions

$$\text{SPACE } 2_i^{\text{LIN}} \subseteq \text{TIME } 2_{i+1}^{\text{LIN}} \subseteq \text{SPACE } 2_{i+1}^{\text{LIN}}$$

is strict; similarly, we know that at least one of the inclusions

$$\text{TIME } 2_i^{\text{LIN}} \subseteq \text{SPACE } 2_i^{\text{LIN}} \subseteq \text{TIME } 2_{i+1}^{\text{LIN}}$$

is strict, and the general opinion is that they all are. Still, no one has ever been able to prove that any particular of the inclusions actually is strict, and we are facing a notoriously hard open problem.

The classes in the alternating space–time hierarchy are defined by imposing *explicit bounds* on a particular machine model, but the classes are *not uniformly defined* as some of the classes are defined by imposing space-bounds whereas others are defined by imposing time-bounds. In contrast, the classes in our T^--hierarchy $\mathcal{G}_0 \subseteq \mathcal{G}_1 \subseteq \mathcal{G}_2 \subseteq \dots$ are *uniformly defined*. They are also defined *without referring to explicit resource bounds*. Thus, the following theorem is interesting and also a bit surprising. How can it be that such the uniformly defined $\mathcal{G}$-hierarchy contains both space and time classes?

Theorem 1 ([21, 22])

We have $\text{SPACE } 2_n^{\text{LIN}} = \mathcal{G}_{2n}$ and $\text{TIME } 2_{n+1}^{\text{LIN}} = \mathcal{G}_{2n+1}$.

Let LOGSPACE denote the set of problems decided by a deterministic Turing machine working in logarithmic space. Let TIME 2_i^{POL} (SPACE 2_i^{POL}) denote the set of problems decidable by a deterministic Turing machine working in time (space) $2_i^{p(|x|)}$ for some polynomial p. Now we have another alternating space–time hierarchy

$$\text{LOGSPACE} \subseteq \text{TIME } 2_0^{\text{POL}} \subseteq \text{SPACE } 2_0^{\text{POL}} \subseteq \text{TIME } 2_1^{\text{POL}} \subseteq \text{SPACE } 2_1^{\text{POL}} \subseteq \text{TIME } 2_2^{\text{POL}} \subseteq \dots$$

analogous to the hierarchy discussed above. The analogous open problems do also emerge. Let $\mathcal{C}_i$, $\mathcal{C}_{i+1}$, $\mathcal{C}_{i+2}$ be three arbitrary consecutive classes in the hierarchy. It is well-known that $\mathcal{C}_i \subset \mathcal{C}_{i+2}$, so at least one of the two inclusions $\mathcal{C}_i \subseteq \mathcal{C}_{i+1}$ and $\mathcal{C}_{i+1} \subseteq \mathcal{C}_{i+2}$ will be strict. Still, for any fixed $j \in \mathbb{N}$, it is an open problem if $\mathcal{C}_j$ is strictly included in $\mathcal{C}_{j+1}$. Note that TIME 2_0^{POL} and SPACE 2_0^{POL} are the classes usually denoted, respectively, P and PSPACE in the literature, so the notorious open problem $\text{LOGSPACE} \overset{?}{\subset} \text{P} \overset{?}{\subset} \text{PSPACE}$ emerges at the bottom of the hierarchy.

The relationship between the two alternating space–time hierarchies is also a bit of a mystery. The only thing known about the relationship between SPACE 2_i^{LIN} and TIME 2_i^{POL} is that the two classes cannot be equal. So, it is known that e.g. LINSPACE $\neq$ P, but it is an open problem if LINSPACE is strictly included in P or if P is strictly included in LINSPACE or if neither of the two classes is included in the other.

T$^-$ can easily be modified into a programming language doing recursion on bit strings in place of recursion on natural numbers. We throw away all the constants $k_0, k_1, k_2, \dots$ and introduce a constant k_α of type ι for each bit string α. The recursor terms needs to be adjusted accordingly, i.e. we need recursor terms of the form $\mathrm{R}_\sigma(G^\sigma, F_0^{\iota,\sigma\to\sigma}, F_1^{\iota,\sigma\to\sigma}, N^\iota)$ and the reduction rules

- $\mathrm{R}_\sigma(G, F_0, F_1, k_\varepsilon) \rhd G$ where ε denotes the empty string
- $\mathrm{R}_\sigma(G, F_0, F_1, k_{b\alpha}) \rhd F_b(k_\alpha, \mathrm{R}_\sigma(G, F_0, F_1, k_\alpha))$ where $b \in \{0,1\}$ and $\alpha \in \{0,1\}^*$.

This modified version of T$^-$ induces a hierarchy $\mathcal{G}^{\mathbf{b}} = \bigcup_{n\in\mathbb{N}} \mathcal{G}_n^{\mathbf{b}}$ which is defined analogously to the hierarchy $\mathcal{G}$.

THEOREM 2 ([21, 22])
We have LOGSPACE $= \mathcal{G}_{\mathbf{0}}^{\mathbf{b}}$ and

$$\text{SPACE } 2_i^{\text{POL}} = \mathcal{G}_{2i+2}^{\mathbf{b}} \quad \text{and} \quad \text{TIME } 2_i^{\text{POL}} = \mathcal{G}_{2i+1}^{\mathbf{b}}$$

Theorems 1 and 2 are proved for the first time in Kristiansen and Voda [21]. The proofs in [21] of the inclusions $\mathcal{G}_{2n} \subseteq \text{SPACE } 2_n^{\text{LIN}}$ and $\mathcal{G}_{2n+1} \subseteq \text{TIME } 2_{n+1}^{\text{LIN}}$ ($\mathcal{G}_{2n}^{\mathbf{b}} \subseteq \text{SPACE } 2_n^{\text{POL}}$ and $\mathcal{G}_{2n+1}^{\mathbf{b}} \subseteq \text{TIME } 2_{n+1}^{\text{POL}}$) are inspired by Goerdt and Seidel's work in finite model theory. See [13, 14]. Essentially different proofs of these inclusions can be found in [22]. The proofs in [22] are based on an adaption of Schwichtenberg's Trade-off Theorem to a complexity-theoretic context. For more on Schwictenberg's Theorem see [28].

1.3 T$^-$ and subrecursive classes

A severely refined version of the T$^-$-hierarchy $\mathcal{G}$ appears in [19]. In [19], we work with sum types in addition to product and arrow types and with a basic type $\mathbf{q}$ in addition to the basic type ι. The type $\mathbf{q}$ is a unit type that contains only one sole element. We define a well ordering $\preceq$ of the types, and then, for each type σ, a class of problems $\mathcal{G}_\sigma$ such that $\mathcal{G}_\rho \subseteq \mathcal{G}_\tau$ iff $\rho \preceq \tau$.

In addition to the complexity classes discussed above, typical subrecursive classes occur in the hierarchy $\mathcal{G} = \bigcup_{\sigma \in \mathrm{TYP}} \mathcal{G}_\sigma$. In contrast to a complexity class, a subrecursive class is defined as the least class containing some initial functions and closed under certain composition and recursion schemes. Some of the schemes might contain explicit bounds, but no machine models are involved. The Grzegorczyk classes $\mathcal{E}^0$, $\mathcal{E}^1$ and $\mathcal{E}^2$ are typical examples. The class $\mathcal{E}^0$ is the closure of u_i^n (projections), S (successor) 0 (zero) under composition and bounded primitive recursion; $\mathcal{E}^1$ and $\mathcal{E}^2$ are defined similarly, but the sets of initial functions are extended with, respectively, $+$ (addition) and $\times$ (multiplication). Now, let $\mathcal{E}_*^0$, $\mathcal{E}_*^1$ and $\mathcal{E}_*^2$, respectively, denote the problems (i.e. the 0–1-valued functions) of the Grzegorczyk classes $\mathcal{E}^0$, $\mathcal{E}^1$ and $\mathcal{E}^2$, and let $\Delta_0^{\mathbb{N}}$ be the set problems definable in Peano Arithmetic by a Δ_0^0 statement.

THEOREM 3 ([19])
We have

- $\Delta_0^{\mathbb{N}} \subseteq \mathcal{G}_{\iota \oplus \mathbf{q}}$
- $\mathcal{G}_{\iota \oplus \iota} = \mathcal{E}_*^0$.
- $\mathcal{G}_{\iota \otimes \iota} = \mathcal{E}_*^1$.
- $\mathcal{G}_{\iota \to (\mathbf{q} \oplus \mathbf{q})} = \mathcal{E}_*^2$.

Thus, T^- embodies yet another notoriously hard open problem seemingly unrelated to the open problems concerning the relationship between space and time classes: *the problem of the small relational Grzegorzyk classes*. It is easy to prove that $\Delta_0^{\mathbb{N}} \subseteq \mathcal{E}_*^0 \subseteq \mathcal{E}_*^1 \subseteq \mathcal{E}_*^2$, but it is not known whether any of the inclusions are strict, indeed it is open if the inclusion $\Delta_0^{\mathbb{N}} \subseteq \mathcal{E}_*^2$ is strict. It is proved in [7] that $\mathcal{E}_*^1 = \mathcal{E}_*^2$ implies $\mathcal{E}_*^0 = \mathcal{E}_*^2$. Furthermore, we know that $\Delta_0^{\mathbb{N}} = \mathcal{E}_*^0$ implies $\Delta_0^{\mathbb{N}} = \mathcal{E}_*^2$ (see [20]). The open problems can be traced back to Grzegorczyk's initial paper [15] from 1953. For more on the Grzegorczyk classes and the rudimentary relations see [8–10, 20, 23, 26, 27].

We expect a lot of more or less natural subrecursive classes to match classes in the refined $\mathcal{G}$-hierarchy, e.g. the hierarchy of subrecursive classes studied in [20].

1.4 The programming language PCF$^-$ *and the hierarchy* $\mathcal{P}$

So far our studies of recursion in higher types and complexity theory seems promising. Our hierarchies are induced by neat and natural fragments of a calculus based on finite types and Gödel's T, and all the classes in the hierarchies are uniformly defined without referring to explicit bounds. Thus, one would not expect the hierarchies to capture such a wide variety of classes, i.e. both time classes, space classes and subrecursive classes. This indicates that a further investigation of the hierarchies might be rewarding, and perhaps shed light upon some of the notoriously hard open problems involving the classes captured by the hierarchies, e.g. maybe some of these problems turn out to be related in some unexpected way.

An obvious way to continue our studies will be to get non-determinism and general recursion into picture. The basic idea needed to introduce non-determinism is very simple. We extend T^- by

$$(M|N) \text{ is a term of type } \sigma \text{ if } M \text{ and } N \text{ are terms of type } \sigma$$

and the two accompanying reduction rules $(M|N) \rhd M$ and $(M|N) \rhd N$. This extension will indeed yield a non-deterministic version of any class in our refined hierarchy, also those classes that we cannot characterize by imposing natural resource bounds on Turing machines. In particular, we have non-deterministic versions of the small Grzegorczyk classes discussed above. How do these non-deterministic Grzegorczyk relates to each other? How do they relate to the deterministic ones?

Non-determinism in T^- and PCF^- is a topic for future research. Some preliminary investigations into the subject are published in [3]. In the current paper, we will introduce general recursion into the picture by extending T^- with fixed-point terms.

DEFINITION 4
The calculus PCF^- is the calculus T^- extended with a *fixed-point term* $\mathrm{Y}_\sigma M$ of type σ for each term M of type $\sigma \to \sigma$ and reduction rules of the form $\mathrm{Y}_\sigma M \rhd M(\mathrm{Y}_\sigma M)$.

The *rank* of a PCF^--term M, written $\mathrm{Rk}(M)$, is the greatest number n such that $n \leq \mathrm{dg}(\sigma)$ for any recursor term $\mathrm{R}_\sigma(\ldots)$ and any fixed-point term $\mathrm{Y}_\sigma(\ldots)$ occurring in M.

We define the hierarchy $\mathcal{P} = \bigcup_{n\in\mathbb{N}} \mathcal{P}_n$ by $A \in \mathcal{P}_n$ iff A is decided by a PCF^--term of rank n.

PCF^- is investigated for the first time in the current paper and is based on the well-known programming language PCF. We will assume the reader is familiar with PCF. See e.g. [31] for an introduction.

PCF^- is essentially PCF without the successor term, but to achieve a smooth and laconic presentation, we have defined PCF^- as a parsimonious extension of T^-. Our main results would still hold if PCF^- were based on a standard version of PCF, i.e. PCF without recursor terms, but with boolean types, predecessor terms, etc.

1.5 Main results and related research

The main result of this article is the following theorem.

THEOREM 4 (Main)
We have $\mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1} = \mathcal{P}_{n+1}$.

When we compare this theorem to Theorem 1, a few questions raise themselves. Why do only time classes appear in the PCF^--hierarchy, whereas the T^--hierarchy contains both time and space classes? Why does the fragment of PCF^-, where the type degree of the fixed-point terms are bounded by $n+1$, correspond exactly to the fragment of T^- where the type degree of the recursor terms are bounded by $2n+1$? To better understand why, we will develop uniform and parallel proofs of the three inclusions

$$\mathrm{SPACE}\ 2^{\mathrm{LIN}}_n \subseteq \mathcal{G}_{2n} \quad \mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1} \subseteq \mathcal{G}_{2n+1} \quad \mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1} \subseteq \mathcal{P}_{n+1}\,.$$

These proofs, which are given in Sections 3.1 and 3.2, will hopefully elucidate the relationship between the computational power of recursion terms and the computational power of fixed-point terms. Besides, the proofs of the inclusions $\mathrm{SPACE}\ 2^{\mathrm{LIN}}_n \subseteq \mathcal{G}_{2n}$ and $\mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1} \subseteq \mathcal{G}_{2n+1}$ are more transparent and direct than those given elsewhere.

The proof of the inclusion $\mathcal{P}_{n+1} \subseteq \mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1}$ calls for domain theory. In Section 2, we develop some domain theory and show how to interpret PCF^--terms in finite domains. This makes it possible to give a transparent proof of the inclusion $\mathcal{P}_{n+1} \subseteq \mathrm{TIME}\ 2^{\mathrm{LIN}}_{n+1}$ in Section 3.3.

There has been much research on relating complexity theory, on the one hand, and logical languages or programming languages, on the other hand: work on tiering by e.g. [5, 29]; work on finite model theory by e.g. Gurevich [16], Immerman [17], Goerdt [14]; work on linear logic by e.g. [11, 12]; work on proof theory by e.g. [6]; work on the typed λ-calculus; and so on. This is research with different starting points, different emphasizes and different motivations. In the end, some of this research might turn out to be more similar than expected, and the author realizes that he should be careful to claim too much originality for some of the research presented in this article.

Work closely related to ours includes work by Mairson, not know, by the author until recently. The techniques used in [24] and [1] to simulate Turing machines in the typed λ-calculus, are similar to the techniques we use to simulate Turing machines in T^- and PCF^-. Furthermore, our main result, i.e. Theorem 4, is really what to expect from reading Jones' paper *'The expressive power of higher order types or, life without CONS'* [18]. Jones' work with a Haskell-like higher ordered programming language where booleans, and lists of booleans, are primitive types. By excluding constructor operations on lists (CONS), he achieve a characterization similar to ours of the hierarchy $\bigcup_{n\in\mathbb{N}} \mathrm{TIME}\ 2_n^{\mathrm{POL}}$. PCF has the same expressive power as Jones' programming language, and excluding constructor operations on lists corresponds to excluding the successor operations on natural numbers. Still, Jones captured a slightly different hierarchy than ours, i.e. $\bigcup_{n\in\mathbb{N}} \mathrm{TIME}\ 2_n^{\mathrm{POL}}$ in contrast to $\bigcup_{n\in\mathbb{N}} \mathrm{TIME}\ 2_n^{\mathrm{LIN}}$. This is due to the choice of basic types: natural numbers versus booleans and lists of booleans. The main difference between the work presented in this article and the work of Jones' is not the choice of programming language or the choice of primitive types, but rather the overall approach: Our work is based on denotational semantics and domain theory whereas Jones' work is based on operational semantics and compiler theory. The most original ingredients of this article are probably the domain theory developed in Section 2 and the application of this theory in Section 3.

2 Interpretations of PCF^--terms

We assume some experience with domain theory. For more on the subject, see e.g. [31].

2.1 Interpretations in finite domains

DEFINITION 5

For any natural number $b>1$, we define the *finite domain* D^b_σ and the binary relation $\sqsubseteq_\sigma$ over D^b_σ recursively over the structure of the type σ. Let $D^b_\iota=\{\perp,0,1,\ldots,b-1\}$ and

$$d\sqsubseteq_\iota e \Leftrightarrow d=\perp \vee d=e\,.$$

Let $D^b_{\rho\to\tau}=\{f:D^b_\rho\to D^b_\tau \mid d\sqsubseteq_\rho e\Rightarrow f(d)\sqsubseteq_\tau f(e)\}$ and

$$d\sqsubseteq_{\rho\to\tau} e \Leftrightarrow \forall a\in D^b_\rho\,[\,d(a)\sqsubseteq_\tau e(a)\,]\,.$$

Let $D^b_{\rho\otimes\tau}=D^b_\rho\times D^b_\tau$ and

$$d\sqsubseteq_{\rho\otimes\tau} e \Leftrightarrow \mathrm{fst}(d)\sqsubseteq_\rho \mathrm{fst}(e)\wedge \mathrm{snd}(d)\sqsubseteq_\tau \mathrm{snd}(e)$$

where fst(x) and snd(x) denote respectively the first and the second component of the pair $x\in D^b_\rho\times D^b_\tau$. We use $\perp_\sigma$ to denote the unique element in D^b_σ such that $\perp_\sigma\sqsubseteq_\sigma d$ holds for any $d\in D^b_\sigma$. Occasionally, we will suppress the subscript and just write $\perp$.

We define the *cardinality of the type σ at base b*, written $|\sigma|_b$, by recursion on the structure of the type σ: $|\iota|_b=b$; $|\rho\otimes\tau|_b=|\rho|_b\times|\tau|_b$; $|\rho\to\tau|_b=|\tau|_b^{|\rho|_b}$.

We define the *domain height of the type σ at base b*, written $\lceil\sigma\rceil_b$, by recursion on the structure of the type σ: $\lceil\iota\rceil_b=1$; $\lceil\rho\otimes\tau\rceil_b=\lceil\rho\rceil_b+\lceil\tau\rceil_b$; $\lceil\rho\to\tau\rceil_b=|\rho|_{b+1}\times\lceil\tau\rceil_b$. ■

LEMMA 1
Let $|D^b_\sigma|$ denote the number of elements in the finite domain D^b_σ. We have $|D^b_\sigma| \leq |\sigma|_{b+1}$.

PROOF. We prove the lemma by induction on the structure of σ. We have $D^b_\iota = \{\perp, 0, 1, \ldots, b-1\}$ and $|\iota|_{b+1} = b+1$ straightaway from the definitions, and thus the lemma holds when $\sigma = \iota$. Assume by induction hypothesis that $|D^b_\rho| \leq |\rho|_{b+1}$ and $|D^b_\tau| \leq |\tau|_{b+1}$. Now, $D^b_{\rho\to\tau}$ is a subset of the functions from D^b_ρ into D^b_τ, and hence we have $|D^b_{\rho\to\tau}| \leq |\tau|_{b+1}^{|\rho|_{b+1}} = |\rho \to \tau|_{b+1}$. Moreover, $D^b_{\rho\otimes\tau} = D^b_\rho \times D^b_\tau$, and hence we have $|D^b_{\rho\otimes\tau}| \leq |\rho|_{b+1} \times |\tau|_{b+1} = |\rho \otimes \tau|_{b+1}$. ■

LEMMA 2
Let $d_0, \ldots, d_m$ be elements in the finite domain D^b_σ such that $d_i \sqsubseteq_\sigma d_{i+1}$ and $d_i \neq d_{i+1}$ for all $i < m$. Then we have $m \leq \lceil\sigma\rceil_b$.

PROOF. Let $d \sqsubset_\sigma d'$ denote that $d \sqsubseteq_\sigma d'$ and $d \neq d'$. We prove the lemma by induction on the structure of σ.

We have $d \sqsubset_\iota d'$ if and only if $d = \perp$ and $d' \in \mathbb{N}$. Hence, if $d_0 \sqsubset_\iota \ldots \sqsubset_\iota d_m$, then $m \leq 1$. Our definitions state that $\lceil\iota\rceil_b = 1$, and thus the lemma holds when $\sigma = \iota$.

Let $\sigma = \rho \otimes \tau$. Assume that

$$\langle a_0, b_0\rangle \sqsubset_{\rho\otimes\tau} \langle a_1, b_1\rangle \sqsubset_{\rho\otimes\tau} \ldots \sqsubset_{\rho\otimes\tau} \langle a_m, b_m\rangle$$

where $m > \lceil\rho\otimes\tau\rceil_b = \lceil\rho\rceil_b + \lceil\tau\rceil_b$. As $\langle a_i, b_i\rangle \neq \langle a_{i+1}, b_{i+1}\rangle$ for $i \in \{0, \ldots, m-1\}$, we have $a_i \neq a_{i+1}$ or $b_i \neq b_{i+1}$ for $i \in \{0, \ldots, m-1\}$. Hence, since $m > \lceil\rho\rceil_b + \lceil\tau\rceil_b$, (at least) one of the two following cases holds

(1) there exists a sub-sequence $a_{i_0}, \ldots a_{i_k}$ of $a_0, \ldots a_m$ such that

$$a_{i_0} \sqsubset_\rho \ldots \sqsubset_\rho a_{i_k} \text{ and } k > \lceil\rho\rceil_b\,.$$

(2) there exists a sub-sequence $b_{i_0}, \ldots b_{i_\ell}$ of $b_0, \ldots b_m$ such that

$$b_{i_0} \sqsubset_\tau \ldots \sqsubset_\tau b_{i_\ell} \text{ and } \ell > \lceil\tau\rceil_b\,.$$

The former case contradicts our induction hypothesis on ρ, the latter case contradicts our induction hypothesis on τ.

Let $\sigma = \rho \to \tau$. Assume that we have

$$d_0 \sqsubset_{\rho\to\tau} \ldots \sqsubset_{\rho\to\tau} d_m$$

where $m > \lceil\rho\to\tau\rceil_b = |\rho|_{b+1} \times \lceil\tau\rceil_b$. For each $i \in \{0, \ldots, m-1\}$, we have $d_i \neq d_{i+1}$, and thus, there exists $a \in D^b_\rho$ such that $d_i(a) \neq d_{i+1}(a)$. Lemma 1 states that the number of elements in D^b_ρ is bounded by $|\rho|_{b+1}$. Hence, as $m > |\rho|_{b+1} \times \lceil\tau\rceil_b$, for at least one element $a \in D^b_\rho$ there exists a sub-sequence $d_{i_0}, \ldots d_{i_k}$ of $d_0, \ldots d_m$ such that

$$d_{i_0}(a) \sqsubset_\tau \ldots \sqsubset_\tau d_{i_k}(a) \text{ and } k > \lceil\tau\rceil_b\,.$$

This contradicts our induction hypothesis on τ. ■

DEFINITION 6

A *(base b) assignment* $\mathcal{A}$ is a total function from the set of PCF$^-$-variables into the set $\bigcup_{\sigma\in\mathrm{TYP}} D^b_\sigma$ such that $\mathcal{A}(x^\sigma)\in D^b_\sigma$, i.e. $\mathcal{A}$ assigns values in the finite domain D^b_σ to variables of type σ. We define the assignment $\mathcal{A}^x_d$ by $\mathcal{A}^x_d(x)=d$ and $\mathcal{A}^x_d(y)=\mathcal{A}(y)$ for any variable y different from x. A PCF$^-$-term M belongs to the fragment PCF^-_b if $i<b$ for each constant k_i occurring in M.

We define the *(base b) interpretation* of the PCF^-_b-term M under the assignment $\mathcal{A}$, written $[\![M]\!]_\mathcal{A}$, by recursion over the structure of M. Let

(1) $[\![k_\ell]\!]_\mathcal{A}=\ell$ for any constant k_ℓ
(2) $[\![x]\!]_\mathcal{A}=\mathcal{A}(x)$ for any variable x
(3) $[\![(MN)]\!]_\mathcal{A}=[\![M]\!]_\mathcal{A}([\![N]\!]_\mathcal{A})$
(4) $[\![\lambda xM]\!]_\mathcal{A}=f$ where the function f is defined by $f(d)=[\![M]\!]_{\mathcal{A}^x_d}$
(5) $[\![\mathbf{fst}M]\!]_\mathcal{A}=\mathrm{fst}([\![M]\!]_\mathcal{A})$
(6) $[\![\mathbf{snd}M]\!]_\mathcal{A}=\mathrm{snd}([\![M]\!]_\mathcal{A})$
(7) $[\![\langle M_1,M_2\rangle]\!]_\mathcal{A}=\langle[\![M_1]\!]_\mathcal{A},[\![M_2]\!]_\mathcal{A}\rangle$
(8)
$$[\![\mathrm{R}_\sigma(G,F,N)]\!]_\mathcal{A}=\begin{cases}\perp_\sigma & \text{if } [\![N]\!]_\mathcal{A}=\perp_\iota\\ f(n) & \text{otherwise}\end{cases}$$
where $n=[\![N]\!]_\mathcal{A}$ and the function f is defined by $f(k+1)=[\![F]\!]_\mathcal{A}(k,f(k))$ and $f(0)=[\![G]\!]_\mathcal{A}$
(9) $[\![(\mathrm{Y}_\sigma M)]\!]_\mathcal{A}=f^k(\perp_\sigma)$ where $k=\lceil\sigma\rceil_b$ and the function f is defined by $f=[\![M]\!]_\mathcal{A}$.

For any closed term M, we have $[\![M]\!]_{\mathcal{A}_0}=[\![M]\!]_{\mathcal{A}_1}$ for any assignments $\mathcal{A}_0,\mathcal{A}_1$, and we will simply write $[\![M]\!]$ when M is closed.

LEMMA 3

If M is a PCF^-_b-term of type σ and $\mathcal{A}$ is a base b assignment, then we have $[\![M]\!]_\mathcal{A}\in D^b_\sigma$.

PROOF. The proof is fairly straightforward. Use induction over the structure of M. We omit the details. ■

LEMMA 4

Let M be a PCF^-_b-term, and let $\mathcal{A}$ be a base b assignment. If $M\rhd N$, then $[\![M]\!]_\mathcal{A}=[\![N]\!]_\mathcal{A}$.

PROOF. The interesting case is the reduction rule $\mathrm{Y}_\sigma M\rhd M(\mathrm{Y}_\sigma M)$. Let $f=[\![M]\!]_\mathcal{A}$. We have $f\in D^b_{\sigma\to\sigma}$, and thus $f^k(\perp)\sqsubseteq_\sigma f^{k+1}(\perp)$ for any $k\in\mathbb{N}$. By Lemma 2, we have $f^k(\perp)=f^{k+1}(\perp)$ for any $k\geq\lceil\sigma\rceil_b$. Hence

$$[\![(\mathrm{Y}_\sigma M)]\!]_\mathcal{A}=f^{\lceil\sigma\rceil_b}(\perp)=ff^{\lceil\sigma\rceil_b}(\perp)=[\![M]\!]_\mathcal{A}([\![\mathrm{Y}_\sigma M]\!]_\mathcal{A})=[\![M(\mathrm{Y}_\sigma M)]\!]_\mathcal{A}\,.$$

The proofs for the remaining reduction rules are standard, and we omit the details. ■

THEOREM 5 (Soundness I)

For any closed PCF^-_b-term M of type ι, we have $[\![M]\!]=n$ whenever $M\overset{\star}{\rhd}k_n$.

PROOF. Assume $M\overset{\star}{\rhd}k_n$, i.e. we have a reduction sequence

$$M\equiv M_0\rhd M_2\rhd\ldots\rhd M_\ell\equiv k_n\,.$$

By Lemma 4, we have $[\![M_i]\!]=[\![M_{i+1}]\!]$ for any $i<\ell$. Thus, we have $[\![M]\!]=n$ since $[\![k_n]\!]=n$. ■

2.2 Interpretations in finite total function spaces

We will now define the interpretation $[\![\![M]\!]\!]_{\mathcal{A}}$ interpreting any PCF_b^--term M as a finite total object.

DEFINITION 7
For any natural number $b>1$, we define the *finite total objects* T_σ^b, and the binary relation $d \preccurlyeq_\sigma t$ where $d \in D_\sigma^b$ and $t \in T_\sigma^b$, recursively over the structure of the type σ. Let $T_\iota^b = \{0,1,\ldots,b-1\}$, and let

$$d \preccurlyeq_\iota t \;\Leftrightarrow\; d=\bot \vee d=t\,.$$

Let $T_{\rho\to\tau}^b$ be the set of all total function from T_ρ^b into T_τ^b, and let

$$d \preccurlyeq_{\rho\to\tau} t \;\Leftrightarrow\; d(\imath) \preccurlyeq_\tau t(\jmath) \text{ for all } \imath \preccurlyeq_\rho \jmath\,.$$

Let $T_{\rho\otimes\tau}^b = T_\rho^b \times T_\tau^b$, and let

$$a \preccurlyeq_{\rho\otimes\tau} b \;\Leftrightarrow\; \mathrm{fst}(a) \preccurlyeq_\rho \mathrm{fst}(a) \wedge \mathrm{snd}(b) \preccurlyeq_\tau \mathrm{snd}(b)\,.$$

Let $\mathcal{A}$ be a (*base b*) assignment. An *$\mathcal{A}$-compatible total assignment* $\hat{\mathcal{A}}$ is a total function from the set of PCF^--variables into the set $\bigcup_{\sigma\in\mathrm{TYP}} T_\sigma^b$ such that $\hat{\mathcal{A}}(x^\sigma) \in T_\sigma^b$ and $\mathcal{A}(x) \preccurlyeq_\sigma \hat{\mathcal{A}}(x)$.

We define the *total (base b) interpretation* of the PCF_b^--term M under the $\mathcal{A}$-compatible assignment $\hat{\mathcal{A}}$, written $[\![M]\!]_{\hat{\mathcal{A}}}$, by recursion over the structure of M. Let

(1) $[\![\![k_\ell]\!]\!]_{\hat{\mathcal{A}}} = \ell$ for any constant k_ℓ
(2) $[\![\![x]\!]\!]_{\hat{\mathcal{A}}} = \hat{\mathcal{A}}(x)$ for any variable x
(3) $[\![\![(MN)]\!]\!]_{\hat{\mathcal{A}}} = [\![\![M]\!]\!]_{\hat{\mathcal{A}}}([\![\![N]\!]\!]_{\hat{\mathcal{A}}})$
(4) $[\![\![\lambda x M]\!]\!]_{\hat{\mathcal{A}}} = f$ where the function f is defined by $f(d) = [\![\![M]\!]\!]_{\hat{\mathcal{A}}_d^x}$
(5) $[\![\![\mathbf{fst} M]\!]\!]_{\hat{\mathcal{A}}} = \mathrm{fst}([\![\![M]\!]\!]_{\hat{\mathcal{A}}})$
(6) $[\![\![\mathbf{snd} M]\!]\!]_{\hat{\mathcal{A}}} = \mathrm{snd}([\![\![M]\!]\!]_{\hat{\mathcal{A}}})$
(7) $[\![\![\langle M_1, M_2\rangle]\!]\!]_{\hat{\mathcal{A}}} = \langle [\![\![M_1]\!]\!]_{\hat{\mathcal{A}}}, [\![\![M_2]\!]\!]_{\hat{\mathcal{A}}}\rangle$
(8) $[\![\![\mathrm{R}_\sigma(G,F,N)]\!]\!]_{\hat{\mathcal{A}}} = f(n)$ where $n = [\![\![N]\!]\!]_{\hat{\mathcal{A}}}$ and the function f is defined by $f(k+1) = [\![\![F]\!]\!]_{\hat{\mathcal{A}}}(k, f(k))$ and $f(0) = [\![\![G]\!]\!]_{\hat{\mathcal{A}}}$
(9) $[\![\![(\mathrm{Y}_\sigma M)]\!]\!]_{\hat{\mathcal{A}}} = f^k(0_\sigma)$ where $k = \lceil\sigma\rceil_b$, the function f is defined by $f = [\![\![M]\!]\!]_{\hat{\mathcal{A}}}$ and 0_σ is some fixed element in T_σ^b. (We explain the role of this fixed element towards the end of this section.)

LEMMA 5
For any type σ and any $t \in T_\sigma^b$, we have $\bot_\sigma \preccurlyeq_\sigma t$.

PROOF. We prove this lemma by induction on structure of σ. The statement follows trivially from the definition of $\preccurlyeq_\sigma$ when $\sigma = \iota$. Assume $\sigma = \rho \to \tau$. Fix an arbitrary t in $T_{\rho\to\tau}^b$. The definition of $\preccurlyeq_{\rho\to\tau}$ states that $\bot_{\rho\to\tau} \preccurlyeq_{\rho\to\tau} t$ holds iff $\bot_{\rho\to\tau}(\imath) \preccurlyeq_\tau t(\jmath)$ for any $\imath \preccurlyeq_\rho \jmath$. By the definition of $\bot_{\rho\to\tau}$, we have $\bot_{\rho\to\tau}(\imath) = \bot_\tau$ for any $\imath \in D_\rho^b$. By the induction hypothesis on τ, we have $\bot_\tau \preccurlyeq_\tau t(\jmath)$ for any $\jmath \in T_\rho^b$. Hence, we have $\bot_{\rho\to\tau}(\imath) = \bot_\tau \preccurlyeq_\tau t(\jmath)$ for any $\imath \in D_\rho^b$ and any $\jmath \in T_\rho^b$, and particularly, we have $\bot_{\rho\to\tau}(\imath) \preccurlyeq_\tau t(\jmath)$ for any $\imath, \jmath$ such that $\imath \preccurlyeq_\rho \jmath$. Thus, the relation $\bot_{\rho\to\tau} \preccurlyeq_{\rho\to\tau} t$ holds. We omit the case when $\sigma = \rho \otimes \tau$. ∎

LEMMA 6
Let $d \in D_{\rho\to\tau}^b$, $d' \in D_\rho^b$, $t \in T_{\rho\to\tau}^b$ and $t' \in T_\rho^b$. If $d \preccurlyeq_{\rho\to\tau} t$ and $d' \preccurlyeq_\rho t'$, then $d(d') \preccurlyeq_\tau t(t')$.

PROOF. This follows straightaway from the definition of $\preccurlyeq_{\rho\to\tau}$. The definition states that $d \preccurlyeq_{\rho\to\tau} t$ iff we have $d(\imath) \preccurlyeq_\tau t(\jmath)$ for any $\imath \preccurlyeq_\rho \jmath$. Hence, we have $d(d') \preccurlyeq_\tau t(t')$ if $d \preccurlyeq_{\rho\to\tau} t$ and $d' \preccurlyeq_\rho t'$. ∎

LEMMA 7
For any assignment $\mathcal{A}$, any *$\mathcal{A}$-compatible total assignment* $\hat{\mathcal{A}}$, and any PCF_b^--term M of type σ, we have $[\![M]\!]_{\mathcal{A}} \preccurlyeq_\sigma [\![[M]\!]]_{\hat{\mathcal{A}}}$.

PROOF. We prove this lemma by induction on the structure of M. *Case* $M \equiv k_n$: we have $[\![k_n]\!]_{\mathcal{A}} \preccurlyeq_\iota [\![[k_n]\!]]_{\hat{\mathcal{A}}}$ by the definition of $\preccurlyeq_\iota$. *Case* $M \equiv x^\sigma$: we have

$$[\![x]\!]_{\mathcal{A}} = \mathcal{A}(x) \preccurlyeq_\sigma \hat{\mathcal{A}}(x) = [\![[x]\!]]_{\hat{\mathcal{A}}}$$

since the assignment $\hat{\mathcal{A}}$ is $\mathcal{A}$-compatible.

Case $M \equiv (M_1^{\rho\to\tau} M_2^\rho)$: we have $[\![M_1]\!]_{\mathcal{A}} \preccurlyeq_{\rho\to\tau} [\![[M_1]\!]]_{\hat{\mathcal{A}}}$ by induction hypothesis on M_1, and we have $[\![M_2]\!]_{\mathcal{A}} \preccurlyeq_{\rho\to\tau} [\![[M_2]\!]]_{\hat{\mathcal{A}}}$ by induction hypothesis on M_2 Hence, we have

$$[\![(M_1M_2)]\!]_{\mathcal{A}} = [\![M_1]\!]_{\mathcal{A}}([\![M_2]\!]_{\mathcal{A}}) \preccurlyeq_\tau [\![[M_1]\!]]_{\hat{\mathcal{A}}}([\![[M_2]\!]]_{\hat{\mathcal{A}}}) = [\![[(M_1M_2)]\!]]_{\hat{\mathcal{A}}}$$

by Lemma 6.

Case $M \equiv \mathrm{R}_\sigma(G,F,N)$: the proof splits into the two cases (i) $[\![N]\!]_{\mathcal{A}} = \bot$ and (ii) $[\![N]\!]_{\mathcal{A}} = n \neq \bot$. In Case (i) we have $[\![\mathrm{R}_\sigma(G,F,N)]\!]_{\mathcal{A}} = \bot$, and thus, $[\![\mathrm{R}_\sigma(G,F,N)]\!]_{\mathcal{A}} \preccurlyeq_\sigma [\![[\mathrm{R}_\sigma(G,F,N)]\!]]_{\hat{\mathcal{A}}}$ by Lemma 5. We turn to Case (ii). By induction hypothesis on N, we have $n = [\![N]\!]_{\mathcal{A}} \preccurlyeq_\sigma [\![[N]\!]]_{\hat{\mathcal{A}}}$. This entails that $[\![[N]\!]]_{\hat{\mathcal{A}}} = n$. By the definitions of $[\![\cdot]\!]_{\mathcal{A}}$ and $[\![[\cdot]\!]]_{\hat{\mathcal{A}}}$, we have

$$[\![\mathrm{R}_\sigma(G,F,N)]\!]_{\mathcal{A}} = [\![F]\!]_{\mathcal{A}}(n-1,[\![F]\!]_{\mathcal{A}}(n-2,\ldots[\![F]\!]_{\mathcal{A}}(0,[\![G]\!]_{\mathcal{A}})))$$

and

$$[\![[\mathrm{R}_\sigma(G,F,N)]\!]]_{\mathcal{A}} = [\![[F]\!]]_{\mathcal{A}}(n-1,[\![[F]\!]]_{\mathcal{A}}(n-2,\ldots[\![[F]\!]]_{\mathcal{A}}(0,[\![[G]\!]]_{\mathcal{A}})))\,.$$

Use a subinduction on n to prove $[\![\mathrm{R}_\sigma(G,F,N)]\!]_{\mathcal{A}} \preccurlyeq_\sigma [\![[\mathrm{R}_\sigma(G,F,N)]\!]]_{\hat{\mathcal{A}}}$. In the case when $n=0$, use the main induction hypothesis on the term G. In the case when $n=m+1$, use the main induction hypothesis on the term F and Lemma 6. We omit the details.

Case $M \equiv \mathrm{Y}_\sigma F$: it suffices to prove that we have

$$[\![F]\!]^k_{\mathcal{A}}(\bot) \preccurlyeq_\sigma [\![[F]\!]]^k_{\hat{\mathcal{A}}}(0_\sigma) \tag{$\dagger$}$$

for any $k \in \mathbb{N}$. We prove ($\dagger$) by subinduction on k. When $k=0$, we have ($\dagger$) by Lemma 5. When $k=m+1$, use the main induction hypothesis on F and Lemma 6. We omit the details.

Case $M \equiv \lambda x^\rho N^\tau$: we will prove that $[\![\lambda xN]\!]_{\mathcal{A}}(d) \preccurlyeq_\tau [\![[\lambda xN]\!]]_{\hat{\mathcal{A}}}(t)$ whenever $d \preccurlyeq_\rho t$ (then we have $[\![\lambda xN]\!]_{\mathcal{A}} \preccurlyeq_{\rho\to\tau} [\![[\lambda xN]\!]]_{\hat{\mathcal{A}}}$ by the definition of $\preccurlyeq_{\rho\to\tau}$). Fix $d \in D^b_\rho$ and $t \in T^b_\rho$ such that $d \preccurlyeq_\rho t$. The assignment $\hat{\mathcal{A}}^x_t$ is $\mathcal{A}^x_d$-compatible since the assignment $\hat{\mathcal{A}}$ is $\mathcal{A}$-compatible. By the induction hypothesis on N, we have

$$[\![\lambda xN]\!]_{\mathcal{A}}(d) = [\![N]\!]_{\mathcal{A}^x_d} \preccurlyeq_\tau [\![[N]\!]]_{\mathcal{A}^x_t} = [\![[\lambda xN]\!]]_{\hat{\mathcal{A}}}(t)\,.$$

This completes the proof in the case when it is of the form $\lambda x^\rho N^\tau$. The proof is straightforward in the cases when M is of the form $\langle M_1, M_2\rangle$, the form $\mathbf{fst}N$ and the form $\mathbf{snd}N$. We omit the details for these cases. ■

THEOREM 6 (Soundness II)
For any closed PCF_b^--term M of type ι, we have $[\![[M]\!]] = n$ whenever $M \overset{\star}{\rhd} k_n$.

PROOF. Assume $M \stackrel{\star}{\rhd} k_n$. By Theorem 5 and Lemma 7, we have $n = [\![M]\!] \preccurlyeq_\iota [\![\![M]\!]\!]$. The definition of the relation $\preccurlyeq_\iota$ states that

$$n \preccurlyeq_\iota [\![\![M]\!]\!] \Leftrightarrow n = \bot \vee n = [\![\![M]\!]\!].$$

Thus, since $n \neq \bot$, we have $[\![\![M]\!]\!] = n$. ■

The interpretation $[\![\cdot]\!]$ is adequate in the sense that we have $M \stackrel{\star}{\rhd} k_n$ whenever $[\![M]\!] = n$ (we will not prove this fact as it is not needed later in the article). The interpretation $[\![\![\cdot]\!]\!]$ is not adequate. For instance, $[\![\![\mathrm{Y}_\iota(\lambda x^\iota x)]\!]\!] = n$ for some $n \in \mathbb{N}$, but it is not the case that $\mathrm{Y}_\iota(\lambda x^\iota x) \stackrel{\star}{\rhd} k_n$. When M does not normalize, the value of $[\![\![M]\!]\!]$ will depend on the choice of 0_σ in the interpretation of the fixed-point terms. The definition states that $[\![\![(\mathrm{Y}_\sigma N)]\!]\!]_{\hat{\mathcal{A}}} = [\![\![N]\!]\!]^k_{\hat{\mathcal{A}}}(0_\sigma)$ where k is a specific natural number and 0_σ is *some* element in T^b_σ. Any choice of $0_\sigma \in T^b_\sigma$ will do in the sense that the validness of Theorem 6 does not depend on the choice.

The interpretation $[\![\cdot]\!]$ is preserved by the rewrite rules, i.e. we have $[\![M]\!]_{\mathcal{A}} = [\![M']\!]_{\mathcal{A}}$ whenever $M \rhd M'$. In contrast, we might have $[\![\![M]\!]\!]_{\mathcal{A}} \neq [\![\![M']\!]\!]_{\mathcal{A}}$ when $M \rhd M'$. This explains why we have to introduce the interpretation into domains before the interpretation into total spaces. The soundness of the interpretation into total spaces cannot be proved directly, i.e. Theorem 6 cannot be proved without invoking Theorem 5.

2.3 Interpretations in finite sets of natural numbers

We will now define the interpretation $\mathbf{val}^{\mathcal{V}}_b(M)$ interpreting any PCF^-_b-term M of type σ as a natural number strictly less than $|\sigma|_b$.

DEFINITION 8
If $a < |\sigma \to \tau|_b$, then a can be viewed as a $|\sigma|_b$-digit number in base $|\tau|_b$, and hence, uniquely written in the form

$$v_0 + v_1|\tau|^1_b + \cdots + v_k|\tau|^k_b$$

where $k = |\sigma|_b - 1$ and $v_j < |\tau|_b$ for $j = 0, \ldots, k$. The numbers $v_0, \ldots, v_k$ are the *digits* in a, and for any $i < |\sigma|_b$, we denote the ith digit of a by $a[i]_b$, i.e. $a[i]_b = v_i$. A *valuation* $\mathcal{V}$ is a total function from the set of variables into $\mathbb{N}$ such that $\mathcal{V}(x) < |\sigma|_b$ for any variable x of type σ. We define the valuation $\mathcal{V}^x_i$ by $\mathcal{V}^x_i(x) = i$ and $\mathcal{V}^x_i(y) = \mathcal{V}(y)$ for any variable y different from x. We define the *value of the term M at the base b under valuation* $\mathcal{V}$, written $\mathbf{val}^{\mathcal{V}}_b(M)$, recursively over the structure of M. Let

(1) $\mathbf{val}^{\mathcal{V}}_b(x) = \mathcal{V}(x)$
(2) $\mathbf{val}^{\mathcal{V}}_b(\lambda x^\sigma M^\tau) = \sum_{i<|\sigma|_b} \mathbf{val}^{\mathcal{V}^x_i}_b(M) \times |\tau|^i_b$
(3) $\mathbf{val}^{\mathcal{V}}_b((MN)) = \mathbf{val}^{\mathcal{V}}_b(M)[\mathbf{val}^{\mathcal{V}}_b(N)]_b$
(4) $\mathbf{val}^{\mathcal{V}}_b(\langle M^\sigma, N^\tau\rangle) = \mathbf{val}^{\mathcal{V}}_b(M) \times |\tau|_b + \mathbf{val}^{\mathcal{V}}_b(N)$
(5) $\mathbf{val}^{\mathcal{V}}_b(\mathbf{fst} M^{\sigma\otimes\tau}) = \mathbf{val}^{\mathcal{V}}_b(M) \ \mathrm{div}\, |\tau|_b$ (integer division)
(6) $\mathbf{val}^{\mathcal{V}}_b(\mathbf{snd} M^{\sigma\otimes\tau}) = \mathbf{val}^{\mathcal{V}}_b(M) \pmod{|\tau|_b}$
(7) $\mathbf{val}^{\mathcal{V}}_b(k_n) = n \pmod{|\iota|_b}$
(8) $\mathbf{val}^{\mathcal{V}}_b(\mathrm{R}_\sigma(G,F,N)) = f(\mathbf{val}^{\mathcal{V}}_b(N))$ where $f(k+1) = (\mathbf{val}^{\mathcal{V}}_b(F)[k]_b)[f(k)]_b$ and $f(0) = \mathbf{val}^{\mathcal{V}}_b(G)$
(9) $\mathbf{val}^{\mathcal{V}}_b(\mathrm{Y}_\sigma M) = f^{\lceil\sigma\rceil_b}(0)$ where $f(x) = \mathbf{val}^{\mathcal{V}}_b(M)[x]_b$

For any closed term M, we have $\mathbf{val}^{\mathcal{V}_0}_b(M) = \mathbf{val}^{\mathcal{V}_1}_b(M)$ for any valuations $\mathcal{V}_0, \mathcal{V}_1$, and we will simply write $\mathbf{val}_b(M)$ when M is closed. ■

THEOREM 7 (Soundness III)
For any closed PCF_b^--term M of type ι, we have $\mathbf{val}_b(M) = n$ whenever $M \overset{\star}{\rhd} k_n$.

PROOF. For any type σ, let $\mathbb{N}_\sigma^b = \{n \mid n < |\sigma|_b\}$. The bijection $\iota_\sigma : T_\sigma^b \to \mathbb{N}_\sigma^b$ is defined by

- $\iota_\iota(x) = x$
- $\iota_{\rho\to\tau}(t) = \sum_{i<|\rho|_b} \iota_\tau(t(\iota_\rho^{-1}(i))) \times |\tau|_b^i$
- $\iota_{\rho\otimes\tau}(t) = \iota_\rho(\mathrm{fst}(t)) \times |\tau|_b + \iota_\tau(\mathrm{snd}(t))$.

We will occasionally suppress the subscript σ of the bijection ι_σ, and we have

(1) $\iota(t(t')) = \iota(t)[\iota(t')]_b$ for any $t \in T_{\rho\to\tau}^b$ and any $t' \in T_\rho^b$
(2) $\iota(\mathrm{fst}\, t) = \iota(t) \mathrm{div} |\tau|_b$ for any $t \in T_{\rho\otimes\tau}^b$
(3) $\iota(\mathrm{snd}\, t) = \iota(t) \pmod{|\tau|_b}$ for any $t \in T_{\rho\otimes\tau}^b$.

It is easy to check that (1), (2) and (3) hold.

CLAIM 1
Let $\mathcal{A}$ be any assignment such that $\mathcal{A}(x) \in T_\sigma^b$ for all x of type σ. Furthermore, let $\mathcal{V}$ be the valuation given by $\mathcal{V}(x^\sigma) = \iota_\sigma(\mathcal{A}(x))$. Then we have $\mathbf{val}_b^{\mathcal{V}}(M) = \iota_\sigma([\![[M]\!]]_{\mathcal{A}})$ for any term M of type σ.

The claim is proved by induction over the structure of the term M. We need (1), (2) and (3) in the tedious, but straightforward, proof. For example, in the case when $M \equiv \lambda x^\rho N^\tau$ we have

$$
\begin{aligned}
\mathbf{val}_b^{\mathcal{V}}(\lambda x^\sigma N^\tau) &= \sum_{i<|\sigma|_b} \mathbf{val}_b^{\mathcal{V}_i^x}(N) \times |\tau|_b^i && \text{def. of } \mathbf{val}_b^{\mathcal{V}} \\
&= \sum_{i<|\sigma|_b} \iota([\![[N]\!]]_{\mathcal{A}^x_{\iota^{-1}(i)}}) \times |\tau|_b^i && \text{ind. hyp.} \\
&= \sum_{i<|\sigma|_b} \iota([\![[\lambda x N]\!]]_{\mathcal{A}}(\iota^{-1}(i))) \times |\tau|_b^i && \text{def. of } [\![[\cdot]\!]] \\
&= \iota([\![[\lambda x N]\!]]_{\mathcal{A}}) && \text{def. of } \iota.
\end{aligned}
$$

We skip the remaining cases in the proof of the claim, and turn to the proof of the very theorem.

Assume $M \overset{\star}{\rhd} k_n$. By Theorem 6, we have $[\![[M]\!]] = n$. By the Claim 1 and the definition of ι_ι, we have $\mathbf{val}_b(M) = \iota_\iota([\![[M]\!]]) = n$. ■

3 Proofs of the main results

The interpretation function $\mathbf{val}_\cdot(\cdot)$ will serve two purposes. One purpose is rather obvious. A closed term $M^{\iota\to\iota}$ is a *total program* if Mk_n normalizes to a constant k_i for any $n \in \mathbb{N}$. We can execute a total program M on input $n \in \mathbb{N}$ by normalizing the term Mk_n. If the term reduces to k_m, the output of the program will be m. Theorem 7 provides an alternative way to execute a total program M on input n: Compute the value $\mathbf{val}_b(Mk_n)$ where

$$b = \max\{i \mid k_i \text{ occurs in the term } Mk_n\} + 1\,.$$

In Section 3.3, we will prove the inclusion $\mathcal{P}_{n+1} \subseteq \mathrm{TIME}\ 2_{n+1}^{\mathrm{LIN}}$ by showing how to compute the value of $\mathbf{val}_x(M)$ on a resource-bounded Turing machine.

The second purpose, the interpretation function will serve, is more subtle. In Section 3.1 we will use $\mathbf{val}_{\cdot}(\cdot)$ to encode arithmetic $\pmod{|\sigma|_{b+1}}$ into T^- and PCF$^-$. For any type σ, we will construct terms

$$0_\sigma : \iota,\ \mathrm{Suc}_\sigma : \iota, \sigma \to \sigma,\ \mathrm{Pred}_\sigma : \iota, \sigma \to \sigma,\ \mathrm{Le}_\sigma : \iota, \sigma, \sigma \to \iota \text{ and } \mathrm{Eq}_\sigma : \iota, \sigma, \sigma \to \iota$$

such that for any closed terms M, M_1 and M_2, we have

(i) $\mathbf{val}_{b+1}(0_\sigma) = 0$
(ii) $\mathbf{val}_{b+1}(\mathrm{Suc}_\sigma(k_b, M)) = \mathbf{val}_{b+1}(M) + 1 \pmod{|\sigma|_{b+1}}$
(iii) $\mathbf{val}_{b+1}(\mathrm{Pred}_\sigma(k_b, M)) = \mathbf{val}_{b+1}(M) - 1 \pmod{|\sigma|_{b+1}}$
(iv)

$$\mathrm{Le}_\sigma(k_b, M_1, M_2) \overset{\star}{\rhd} \begin{cases} k_0 & \text{if } \mathbf{val}_{b+1}(M_1) \le \mathbf{val}_{b+1}(M_2) \\ k_1 & \text{otherwise} \end{cases}$$

(v)

$$\mathrm{Eq}_\sigma(k_b, M_1, M_2) \overset{\star}{\rhd} \begin{cases} k_0 & \text{if } \mathbf{val}_{b+1}(M_1) = \mathbf{val}_{b+1}(M_2) \\ k_1 & \text{otherwise.} \end{cases}$$

These arithmetical terms become helpful in Section 3.2 where we prove the inclusion TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{P}_{n+1}$ and the inclusions SPACE $2^{\text{LIN}}_n \subseteq \mathcal{G}_{2n}$ and TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{G}_{2n+1}$.

3.1 *Arithmetic in* T$^-$ *and* PCF$^-$

Variants of the next few lemmas can also be found in [21], but we will repeat the core of the proofs here.

LEMMA 8 (Conditionals)
For any type σ there exists a T$^-$-term

$$\mathrm{Cond}_\sigma : \iota, \sigma, \sigma \to \sigma$$

of rank 0 such that $\mathrm{Cond}_\sigma(k_0, M_1, M_2) = M_1$ and $\mathrm{Cond}_\sigma(k_n, M_1, M_2) = M_2$ when $n > 0$.

PROOF. We prove the lemma by induction on the structure of σ.

Let $\mathrm{Cond}_\iota \equiv \lambda x^\iota y^\iota z^\iota . \mathrm{R}_\iota(y, \lambda u^\iota v^\iota . z, x)$. It is easy to see that the term Cond_ι possesses the required properties.

Assume $\sigma = \tau \to \rho$. Let $\mathrm{Cond}_\sigma \equiv \lambda x^\iota X^\sigma Y^\sigma z^\tau . \mathrm{Cond}_\rho(x, Xz, Yz)$. Then, by the induction hypothesis, we have

$$\mathrm{Cond}_\sigma(k_0, F, G) = \lambda z^\tau . \mathrm{Cond}_\rho(k_0, Fz, Gz) = \lambda z^\tau . Fz = F\,.$$

The rightmost equality holds since the calculus permits η-reduction. By a similar argument, we have $\mathrm{Cond}_\sigma(k_{n+1}, F, G) = G$.

Assume $\sigma = \tau \times \rho$. Let

$$\mathrm{Cond}_\sigma \equiv \lambda x^\iota X^\sigma Y^\sigma . \langle \mathrm{Cond}_\tau(x, \mathbf{fst}\ X, \mathbf{fst}\ Y), \mathrm{Cond}_\rho(x, \mathbf{snd}\ X, \mathbf{snd}\ Y) \rangle\,.$$

Use the η-equality $\langle \mathbf{fst} X, \mathbf{snd} X \rangle = X$ to prove that the term Cond_σ possesses the required properties. We omit the details.

Only recursor terms of type ι occur in Cond_σ, and thus we have $\mathrm{Rk}(\mathrm{Cond}_\sigma) = 0$. ■

LEMMA 9 (Long Iterations in T^-)
For all types σ and τ there exists a T^--term $\mathrm{It}^\sigma_\tau : (\iota, \tau \to \tau, \tau) \to \tau$ such that $\mathrm{It}^\sigma_\tau(k_b, F, G) = F^{|\sigma|_{b+1}} G$. Furthermore, we have $\mathrm{Rk}(\mathrm{It}^\sigma_\tau) = \mathrm{dg}(\sigma) + \mathrm{dg}(\tau)$. (We will call It^σ_τ an iterator.)

PROOF. We prove the lemma by induction on the structure of σ.

Assume $\sigma = \iota$. Let $\mathrm{It}^\iota_\tau \equiv \lambda n^\iota Y^{\tau\to\tau} X^\tau . \mathrm{R}_\tau(Y(X), \lambda x^\iota . Y, n)$. Obviously, we have $\mathrm{Rk}(\mathrm{It}^\iota_\tau) = \mathrm{dg}(\iota) + \mathrm{dg}(\tau)$, and it is straightforward to prove by induction on b that $\mathrm{It}^\sigma_\tau(k_b, F, G) = F^{b+1}(G)$. Thus, the lemma holds when $\sigma = \iota$ since $|\iota|_{b+1} = b+1$.

Assume $\sigma = \sigma_1 \to \sigma_2$. Let $\mathrm{It}^\sigma_\tau \equiv \lambda x^\iota Y^{\tau\to\tau} X^\tau . (\mathrm{It}^{\sigma_1}_{\tau\to\tau}(x, \mathrm{It}^{\sigma_2}_\tau(x), Y) X)$. We have

$$\begin{aligned}
\mathrm{Rk}(\mathrm{It}^\sigma_\tau) &= \max(\mathrm{Rk}(\mathrm{It}^{\sigma_1}_{\tau\to\tau}), \mathrm{Rk}(\mathrm{It}^{\sigma_2}_\tau)) && \text{def. of Rk}\\
&= \max(\mathrm{dg}(\sigma_1) + \mathrm{dg}(\tau\to\tau), \mathrm{dg}(\sigma_2) + \mathrm{dg}(\tau)) && \text{ind. hyp.}\\
&= \max(\mathrm{dg}(\sigma_1) + \mathrm{dg}(\tau) + 1, \mathrm{dg}(\sigma_2) + \mathrm{dg}(\tau)) && \text{def. of dg}\\
&= \max(\mathrm{dg}(\sigma_1) + 1, \mathrm{dg}(\sigma_2)) + \mathrm{dg}(\tau) &&\\
&= \mathrm{dg}(\sigma) + \mathrm{dg}(\tau). && \text{def. of dg}
\end{aligned}$$

So, the iterator has the right rank. We will now prove that we indeed have $\mathrm{It}^\sigma_\tau(k_b, F, G) = F^{|\sigma|_{b+1}} G$. Let $A \equiv (\mathrm{It}^{\sigma_2}_\tau k_b)$. We prove by induction on k that $(A^k F)G = F^{|\sigma_2|^k_{b+1}} G$ (*). We have $(A^0 F) = F$, and hence $(A^0 F)G = F^{|\sigma_2|^0_{b+1}} G$. Moreover,

$$(A^{k+1}F)G = (A(A^k F))G = \mathrm{It}^{\sigma_2}_\tau(k_b, A^k F, G) = (A^k F)^{|\sigma_2|_{b+1}} G = F^{|\sigma_2|^{k+1}_{b+1}} G.$$

The two last equalities hold, respectively, by induction hypothesis on σ_2 and by induction hypothesis on k. This proves (*). Furthermore, we have

$$\begin{aligned}
\mathrm{It}^\sigma_\tau(k_b, F, G) &= \mathrm{It}^{\sigma_1}_{\tau\to\tau}(k_b, (\mathrm{It}^{\sigma_2}_\tau k_b), F) G && \text{def. of } \mathrm{It}^\sigma_\tau\\
&= (\mathrm{It}^{\sigma_2}_\tau k_b)^{|\sigma_1|_{b+1}} F) G && \text{ind. hyp. on } \sigma_1\\
&= F^{|\sigma_2|^{|\sigma_1|_{b+1}}_{b+1}} G && (*)\\
&= F^{|\sigma|_{b+1}} G. && \text{def. of } |\sigma|_{b+1}
\end{aligned}$$

When $\sigma = \sigma_1 \times \sigma_2$, let $\mathrm{It}^\sigma_\tau \equiv \lambda x^\iota Y^{\tau\to\tau} X^\tau . \mathrm{It}^{\sigma_1}_\tau(x, \mathrm{It}^{\sigma_2}_\tau(x, Y), X)$ and the lemma holds. We omit the details. ■

LEMMA 10 (Basic functions)
The following number-theoretic functions can be defined by T^--terms of rank 0. (i) $x \dot{-} y$ (modified subtraction); (ii) c where $c(x, y_1, y_2) = y_1$ if $x = 0$; and $c(x, y_1, y_2) = y_2$ if $x \neq 0$. (iii) f where $f(x, m) = x + 1 \pmod{m+1}$ for $x \leq m$.

PROOF. The constant function 0 is defined by the initial T^--term k_0. The projection function $u^n_i(x_1, \ldots, x_n) = x_i$ is defined by the T^--term $\lambda x_1 \ldots x_n . x_i$ (for any fixed $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$). The set of functions defined by T^--terms of rank 0 is obviously closed under composition and primitive recursion. Hence, it is sufficient to assure that the functions in the lemma can be defined from projections and the constant 0 by composition and primitive recursion.

We can define the predecessor function P by the primitive recursion $P(0) = 0$ and $P(y+1) = u^2_1(y, P(y))$, and then (i) holds since $x \dot{-} 0 = 0$ and $x \dot{-} (y+1) = P(x \dot{-} y)$. Furthermore, (ii) holds since the function c is defined by the primitive recursion $c(0, y_1, y_2) = u^2_1(y_1, y_2)$ and $c(x+1, y_1, y_2) = u^4_2(y_1, y_2, c(x, y_1, y_2))$. Finally, (iii) holds since $c(m \dot{-} x, 0, m \dot{-} ((m \dot{-} x) \dot{-} 1)) = x + 1 \pmod{m}$ for $x \leq m$. ■

LEMMA 11 (Arithmetic in T^{-})
For any type σ there exists T^{-}-terms

$$0_\sigma : \iota,\ \mathrm{Suc}_\sigma : \iota,\sigma \to \sigma,\ \mathrm{Pred}_\sigma : \iota,\sigma \to \sigma,\ \mathrm{Le}_\sigma : \iota,\sigma,\sigma \to \iota \text{ and } \mathrm{Eq}_\sigma : \iota,\sigma,\sigma \to \iota$$

of rank $2\mathrm{dg}(\sigma) \dot{-} 2$ such that (i), (ii), (iii), (iv) and (v) at page 294 hold.

PROOF. Let $0_\iota \equiv k_0$; $0_{\pi\otimes\tau} \equiv \langle 0_\pi, 0_\tau\rangle$; and $0_{\pi\to\tau} \equiv \lambda x^\pi 0_\tau$. Obviously, we have $\mathbf{val}_{b+1}(0_\sigma) = 0$ and $\mathrm{Rk}(0_\sigma) = 0 \leq 2\mathrm{dg}(\sigma) \dot{-} 2$ for any σ. Thus, (i) holds.

We will define Suc_σ, Le_σ and Eq_σ in parallel recursively over the structure of σ. We omit the definition of Pred_σ as this definition is very similar to the definition of Suc_σ.

Let $\sigma = \iota$. Given Lemma 10, it is easy to see that we can define the required terms in this case. We omit the details.

Let $\sigma = \pi \to \tau$. We define F by

$$\begin{aligned} F \equiv \lambda b^\iota X^\sigma Y^\sigma z^{\iota\times\pi}.\mathrm{Cond}_{\iota\times\pi}(&\mathrm{Eq}_\tau(b, X(\mathbf{snd} z), Y(\mathbf{snd} z)),\\ &\langle \mathbf{fst} z, \mathrm{Suc}_\pi(b, \mathbf{snd} z)\rangle, \mathrm{Cond}_{\iota\times\pi}(\mathrm{Le}_\tau(b, X(\mathbf{snd} z), Y(\mathbf{snd} z)),\\ &\qquad \langle k_0, \mathrm{Suc}_\pi(b, \mathbf{snd} z)\rangle, \langle k_1, \mathrm{Suc}_\pi(b, \mathbf{snd} z)\rangle)). \end{aligned}$$

By the induction hypothesis, we have

$$F(k_b, M, N, \langle i, j\rangle) = \begin{cases} \langle i, \mathrm{Suc}_\pi(j)\rangle & \text{if } \mathbf{val}_{b+1}(M)[\mathbf{val}_{b+1}(j)]_{b+1} = \mathbf{val}_{b+1}(N)[\mathbf{val}_{b+1}(j)]_{b+1}\\ \langle k_0, \mathrm{Suc}_\pi(j)\rangle & \text{if } \mathbf{val}_{b+1}(M)[\mathbf{val}_{b+1}(j)]_{b+1} < \mathbf{val}_{b+1}(N)[\mathbf{val}_{b+1}(j)]_{b+1}\\ \langle k_1, \mathrm{Suc}_\pi(j)\rangle & \text{otherwise.} \end{cases}$$

Thus, we have

$$\mathbf{fst}(F(k_b, M, N)^{|\pi|_{b+1}}(\langle k_0, 0_\pi\rangle)) = \begin{cases} k_0 & \text{if } \mathbf{val}_{b+1}(M) \leq \mathbf{val}_{b+1}(N)\\ k_1 & \text{otherwise.} \end{cases}$$

and then (iv) holds when

$$\mathrm{Le}_\sigma \equiv \lambda bXY.\mathbf{fst}\,\mathrm{It}^\pi_{\iota\times\pi}(b, F(b, X, Y), \langle k_0, 0_\pi\rangle). \tag{†}$$

Now, let $\mathrm{Eq}_\sigma \equiv \lambda bXY.\mathrm{Cond}_\iota(\mathrm{Le}_\sigma(b, X, Y), \mathrm{Cond}_\iota(\mathrm{Le}_\sigma(b, Y, X), k_0, k_1), k_1)$ and (v) holds. We will now argue that Le_σ and Eq_σ have the required rank. First, we note that

$$\mathrm{Rk}(F) \leq \max(2\mathrm{dg}(\pi) \dot{-} 2, 2\mathrm{dg}(\tau) \dot{-} 2) \tag{*}$$

follows from Lemma 8 and the induction hypothesis. Furthermore, we have

$$\begin{aligned} \mathrm{Rk}(\mathrm{Eq}_\sigma) &= \mathrm{Rk}(\mathrm{Le}_\sigma) && \text{def. of Rk, def. of } \mathrm{Eq}_\sigma\\ &= \max(\mathrm{Rk}(F), \mathrm{Rk}(\mathrm{It}^\pi_{\iota\times\pi})) && \text{def. of Rk, def. of } \mathrm{Le}_\sigma\\ &\leq \max(\mathrm{Rk}(F), \mathrm{dg}(\pi) + \mathrm{dg}(\iota\times\pi)) && \text{Lemma 9}\\ &\leq \max(2\mathrm{dg}(\pi) \dot{-} 2, 2\mathrm{dg}(\tau) \dot{-} 2, 2\mathrm{dg}(\pi)) && (*)\\ &= \max(2\mathrm{dg}(\pi), 2\mathrm{dg}(\tau) \dot{-} 2)\\ &= \max(2\mathrm{dg}(\pi) + 2, 2\mathrm{dg}(\tau)) \dot{-} 2\\ &= 2\max(\mathrm{dg}(\pi) + 1, \mathrm{dg}(\tau)) \dot{-} 2\\ &= 2\mathrm{dg}(\sigma) \dot{-} 2. && \text{def. of dg, } \sigma = \pi \to \tau \end{aligned}$$

Thus, Eq_σ and Le_σ have the required rank. Next we define Suc_σ. Any number $a < |\sigma|_b$ can be uniquely written in the form $a = v_0|\tau|_b^0 + v_1|\tau|_b^1 + \cdots + v_k|\tau|_b^k$ where $k = |\pi|_b - 1$ and $v_j < |\tau|_b$ for $j = 1, \ldots, k$. There exists $i \leq k$ such that

$$a+1 = v_0'|\tau|_b^0 + \cdots + v_i'|\tau|_b^i + v_{i+1}|\tau|_b^{i+1} + \cdots + v_k|\tau|_b^k \pmod{|\sigma|_b}$$

where $v_j' = v_j + 1 \pmod{|\tau|_b}$ for $j = 0, \ldots, i$. We call such an i the *carry border* of the number a. Let

$$\mathrm{C}_\sigma \equiv \lambda bX.\mathbf{snd}\,\mathrm{It}^{\pi}_{\iota\times\pi}(b, G(b,X), \langle k_0, 0_\pi\rangle) \tag{‡}$$

where

$$G \equiv \lambda b^\iota X^{\pi\to\tau} z^{\iota\times\pi}.\mathrm{Cond}_\iota(\mathbf{fst}z, \mathrm{Cond}_{\iota\times\pi}(\mathrm{Eq}_\tau(b, \mathrm{Suc}_\tau(X(\mathbf{snd}z)), 0_\tau), \\ \langle k_0, \mathrm{Suc}_\pi(\mathbf{snd}z)\rangle, \langle k_1, \mathrm{Suc}_\pi(\mathbf{snd}z)\rangle), \langle k_1, \mathrm{Suc}_\pi(\mathbf{snd}z)\rangle).$$

Now, $\mathbf{val}_{b+1}(\mathrm{C}_\sigma(k_b, M))$ equals the carry border of $\mathbf{val}_{b+1}(M)$ when $M{:}\sigma$ is a closed term. Let

$$\mathrm{Suc}_\sigma \equiv \lambda b^\iota X^\sigma i^\pi.\mathrm{Cond}_\tau(\mathrm{Le}_\pi(b, i, \mathrm{C}_\sigma(b,X)), \mathrm{Suc}_\tau(X(i)), X(i))$$

and (i) holds. An argument similar to the one showing that the ranks of Eq_σ and Le_σ are bounded by $2\mathrm{dg}(\sigma)\dot{-}2$, will show that also the rank of C_σ is bounded by $2\mathrm{dg}(\sigma)\dot{-}2$. The rank of Suc_σ equals the rank of C_σ.

Let $\sigma = \pi\times\tau$. Define Suc_σ such that

$$\mathrm{Suc}_\sigma(b, \langle F, G\rangle) = \begin{cases} \langle \mathrm{Suc}_\pi(F), \mathrm{Suc}_\tau(G)\rangle & \text{if } \mathrm{Eq}_\tau(b, \mathrm{Suc}_\tau(G)) = 0_\tau \\ \langle F, \mathrm{Suc}_\tau(G)\rangle & \text{otherwise.} \end{cases}$$

Define Le_σ such that $\mathrm{Le}_\sigma(b, \langle F, G\rangle, \langle F', G'\rangle) = k_0$ iff

$$(\mathrm{Le}_\pi(b,F,F') = k_0 \wedge \mathrm{Eq}_\pi(b,F,F') \neq k_0) \vee \\ (\mathrm{Le}_\pi(b,G,G') = k_0 \wedge \mathrm{Eq}_\pi(b,F,F') = k_0)$$

Define Eq_σ as above. It is easy to construct the required terms, and we skip the details. ■

LEMMA 12 (Arithmetic in PCF$^-$)
For any type σ there exists PCF$^-$-terms

$$0_\sigma{:}\iota,\ \mathrm{Suc}_\sigma{:}\iota,\sigma\to\sigma,\ \mathrm{Pred}_\sigma{:}\iota,\sigma\to\sigma,\ \mathrm{Le}_\sigma{:}\iota,\sigma,\sigma\to\iota \text{ and } \mathrm{Eq}_\sigma{:}\iota,\sigma,\sigma\to\iota$$

of rank $\mathrm{dg}(\sigma)$ such that (i), (ii), (iii), (iv) and (v) at page 294 hold.

PROOF. This proof is nearly identical to the proof of Lemma 11. We define the terms Suc_σ, Pred_σ, Le_σ and Eq_σ as we do in the proof of Lemma 11, i.e. in parallel recursively over the structure of σ, but now we will use fixed-point terms in place of the iterators. This will reduce the ranks of the terms we are defining.

When $\sigma = \pi \to \tau$, the statement marked (†) in the proof of Lemma 11 defines the T$^-$-term Le_σ by $\mathrm{Le}_\sigma \equiv \lambda bXY.\mathbf{fst}\,\mathrm{It}^{\pi}_{\iota\times\pi}(b, F(b,X,Y), \langle k_0, 0_\pi\rangle)$ where F is a term such that

$$\mathbf{fst}(F(k_b,M,N)^{|\pi|_{b+1}}(\langle k_0,0_\pi\rangle)) = \begin{cases} k_0 & \text{if } \mathbf{val}_{b+1}(M) \le \mathbf{val}_{b+1}(N) \\ k_1 & \text{otherwise.} \end{cases} \tag{*}$$

and

$$\mathrm{Rk}(F) = \max(\mathrm{Rk}(\mathrm{Eq}_\tau), \mathrm{Rk}(\mathrm{Suc}_\pi)) \tag{**}$$

Given a term F with these properties, we can define a PCF$^-$-term Le_σ by applying the fixed-point term $(\mathrm{Y}_{\pi\to\iota\otimes\pi}A)$ where

$$A \equiv \lambda U^{\pi\to\iota\otimes\pi} W^{\pi}.\mathrm{Cond}_{\iota\otimes\pi}(\,\mathrm{Eq}_\pi(b, \mathrm{Suc}_\pi(b,W), 0_\pi)\,, \langle k_0, 0_\pi\rangle\,, F(b,X,Y)U(\mathrm{Suc}_\pi(b,W))\,).$$

Then we have

$$(\mathrm{Y}_{\pi\to\iota\otimes\pi}A)0_\pi = F(k_b,X,Y)^{|\pi|_{b+1}}(\langle k_0,0_\pi\rangle)\,.$$

Now, let Le_σ be the PCF$^-$-term given by $\mathrm{Le}_\sigma \equiv \lambda bXY.\mathbf{fst}((\mathrm{Y}_{\pi\to\iota\otimes\pi}A)0_\pi)$. It follows by (*) and induction hypothesis on Eq_π and Suc_π that clause (iv) of our lemma holds. By inspecting the construction of Le_σ, we see that

$$\begin{aligned}\mathrm{Rk}(\mathrm{Le}_\sigma) = \max(\mathrm{Rk}(F), \mathrm{Rk}(\mathrm{Eq}_\pi), \mathrm{Rk}(\mathrm{Suc}_\pi), \mathrm{dg}(\pi \to \iota\otimes\pi)) &\overset{(**)}{=} \\ \max(\mathrm{Rk}(\mathrm{Eq}_\tau), \mathrm{Rk}(\mathrm{Eq}_\pi), \mathrm{Rk}(\mathrm{Suc}_\pi), \mathrm{dg}(\pi \to \iota\otimes\pi))&\end{aligned} \tag{†}$$

To verify that Le_σ has the required rank, we assume by induction hypothesis that

$$\mathrm{Rk}(\mathrm{Eq}_\tau) = \mathrm{dg}(\tau) \text{ and } \mathrm{Rk}(\mathrm{Eq}_\pi) = \mathrm{Rk}(\mathrm{Suc}_\pi) = \mathrm{dg}(\pi)\,. \tag{‡}$$

Then we have

$$\begin{aligned}\mathrm{Rk}(\mathrm{Le}_\sigma) &= \max(\mathrm{Rk}(\mathrm{Eq}_\tau), \mathrm{Rk}(\mathrm{Eq}_\pi), \mathrm{Rk}(\mathrm{Suc}_\pi), \mathrm{dg}(\pi \to \iota\otimes\pi)) && (†) \\ &= \max(\mathrm{Rk}(\mathrm{Eq}_\tau), \mathrm{Rk}(\mathrm{Suc}_\pi), \mathrm{dg}(\pi)+1) && \text{def. of dg} \\ &= \max(\mathrm{dg}(\tau), \mathrm{dg}(\pi), \mathrm{dg}(\pi)+1) && (‡) \\ &= \max(\mathrm{dg}(\pi)+1, \mathrm{dg}(\tau)) \\ &= \mathrm{dg}(\sigma)\,. && \text{def. of dg}\end{aligned}$$

Along this line, we can also find a PCF$^-$-term Suc_σ satisfying the lemma by eliminating the iterator $\mathrm{It}^{\pi}_{\iota\times\pi}$ from the formula marked (‡) in the proof of Lemma 11. The definition of Pred_σ is similar to the definition of Suc_σ, and Eq_σ is defined straightforwardly from Le_σ. ■

3.2 *Simulating turing machines in* T$^-$ *and* PCF$^-$

THEOREM 8

Let σ and ϕ be types, and let $\mathfrak{m}$ be a Turing machine running in space s and time t where $s(|x|) < |\sigma|_{\max(x,2)}$ and $t(|x|) < |\pi|_{\max(x,2)}$. (i) There exists a T$^-$-term $\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}$ of rank $\mathrm{dg}(\pi)+\mathrm{dg}(\sigma)+1$ such that $\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_0$ if $\mathfrak{m}$ accepts x; and $\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_1$ if $\mathfrak{m}$ rejects x. (ii) There exists a PCF$^-$-term

$\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}$ of rank $\max(\mathrm{dg}(\pi),\mathrm{dg}(\sigma))+1$ such that $\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_0$ if $\mathfrak{m}$ accepts x; and $\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_1$ if $\mathfrak{m}$ rejects x.

Proof. Let $\{a_0,\dots,a_\iota\}$ and $\{q_0,\dots,q_J\}$ be, respectively, $\mathfrak{m}$'s alphabet and $\mathfrak{m}$'s set of states. Let q_0 and q_1 be, respectively, the accept and reject state. We can w.l.o.g. assume that the input x is greater than 1, and thus, we can represent a configuration of $\mathfrak{m}$ by a closed term $\langle S^\iota, \langle T^{\sigma\to\iota}, H^\sigma\rangle\rangle$ of type $\xi = \iota\otimes((\sigma\to\iota)\otimes\sigma)$ where S represents the current state, T represents the tape and H the position of the head.

- $\mathbf{val}_x(S) = i$ iff q_i is the current state;
- $\mathbf{val}_x(T)[i]_x = j$ iff the ith cell of the tape contains a_j;
- $\mathbf{val}_x(H) = i$ iff the head scans the ith cell of the tape.

Furthermore, we can use the functionals given by Lemmas 11 and 12 to simulate the execution of $\mathfrak{m}$ on input x. The functionals $\mathrm{Pred}_\sigma(k_x){:}\sigma\to\sigma$ and $\mathrm{Suc}_\sigma(k_x){:}\sigma\to\sigma$ will move the head back and forth, and the functional

$$\mathrm{Md} \equiv \lambda F^{\sigma\to\iota} X^\sigma V^\iota Y^\sigma.\mathrm{Cond}_\iota(\mathrm{Eq}_\sigma(k_x, X, Y), V, F(Y))$$

will modify the tape, e.g. $\mathrm{Md}(T,H,k_{17})$ writes the symbol a_{17} in the scanned cell. We construct terms $\mathrm{Step}_\sigma{:}\iota,\xi\to\xi$ and $\mathrm{Init}_\sigma{:}\iota\to\xi$ such that $\mathrm{Init}_\sigma(k_x)$ represents the initial configuration of $\mathfrak{m}$ on input x, and $\mathrm{Step}_\sigma(k_x,\mathrm{Init}_\sigma(k_x))$ represents the configuration after one transition, $\mathrm{Step}_\sigma(k_x,\mathrm{Step}_\sigma(k_x,\mathrm{Init}_\sigma(k_x)))$ represents the configuration after two transitions and so on. We construct Step_σ such that $\mathrm{Step}_\sigma(k_x,C) = C$ when C represents a halt configuration.

It is easy to see that these terms can be constructed such that $\mathrm{Rk}(\mathrm{Step}_\sigma) = \mathrm{Rk}(\mathrm{Init}_\sigma) = \mathrm{Rk}(\mathrm{Suc}_\sigma) = \mathrm{Rk}(\mathrm{Pred}_\sigma) = \mathrm{Rk}(\mathrm{Eq}_\sigma)$. Thus, by Lemma 11, Step_σ and Init_σ are of rank $2\mathrm{dg}(\sigma)\dot{-}2$ if we are working in T^-, and by Lemma 12, Step_σ and Init_σ are of rank $\mathrm{dg}(\sigma)$ if we are working in PCF^-.

To execute sufficiently many of $\mathfrak{m}$'s transitions in T^-, we apply the iterator It^σ_ξ, i.e. the T^--term given by Lemma 9. We have

$$\mathrm{It}^\pi_\xi(k_x, \mathrm{Step}_\sigma(k_x), \mathrm{Init}_\sigma(k_x)) \;=\; \mathrm{Step}_\sigma(k_x)^{|\pi|_x}(\mathrm{Init}_\sigma(k_x))$$

and thus, the term simulates since $\mathfrak{m}$ runs in time $|\pi|_x$. Let

$$\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma} \equiv \lambda y^\iota \mathbf{fst}\mathrm{It}^\pi_\xi(y, \mathrm{Step}_\sigma(y), \mathrm{Init}_\sigma(y))\,.$$

Then, $\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_0$ if $\mathfrak{m}$ accepts x, and $\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_1$ if $\mathfrak{m}$ rejects x. Furthermore, since $\mathrm{dg}(\xi) = \mathrm{dg}(\sigma)+1$, we have

$$\mathrm{Rk}(\mathrm{T}^{\mathfrak{m}}_{\pi,\sigma}) = \mathrm{Rk}(\mathrm{It}^\pi_\xi) = \mathrm{dg}(\pi)+\mathrm{dg}(\xi) = \mathrm{dg}(\pi)+\mathrm{dg}(\sigma)+1\,.$$

This proves (i).

Next, assume we are working in PCF^-. Let $\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma} \equiv \lambda y^\iota \mathbf{fst}((\mathrm{Y}_{\pi\to\xi}P)0_\pi)$ where

$$\begin{aligned} P \equiv \lambda X^{\pi\to\xi} Z^\pi.\mathrm{Cond}_\xi(&\mathrm{Eq}_\pi(y^\iota, \mathrm{Suc}_\pi(y^\iota, Z), 0_\pi),\\ &\mathrm{Init}_\sigma(y^\iota), \mathrm{Step}_\sigma(y^\iota, X\mathrm{Suc}_\pi(y^\iota, Z))))\,. \end{aligned}$$

Then, $\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_0$ if $\mathfrak{m}$ accepts x, and $\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}(k_x) \overset{\star}{\rhd} k_1$ if $\mathfrak{m}$ rejects x. Furthermore,

$$\mathrm{Rk}(\mathrm{PCF}^{\mathfrak{m}}_{\pi,\sigma}) = \mathrm{dg}(\pi \to \xi) = \max(\mathrm{dg}(\pi)+1, \mathrm{dg}(\xi)) =$$
$$\max(\mathrm{dg}(\pi)+1, \mathrm{dg}(\sigma \to \iota)) = \max(\mathrm{dg}(\pi), \mathrm{dg}(\sigma))+1\,.$$

This proves (ii) ■

LEMMA 13
For any $n,k \in \mathbb{N}$, there exists a type σ of degree n such that $2^{k|x|}_{n+1} \le |\sigma|_{\max(x,2)}$.

PROOF. The number of bits required to represent x in binary notation, written $|x|$, is bounded by $\log_2(2x+2)$, and hence we have

$$2^{k|x|}_{n+1} \le 2^{k\log_2(2x+2)}_{n+1} \le 2^{\log_2(2x+2)^k}_{n+1} \le 2^{(2x+2)^k}_{n}\,.$$

We prove by induction on n that there exists a type σ of degree n such that $2^{(2x+2)^k}_{n} \le |\sigma|_{\max(x,2)}$. When $n=0$, let $\sigma = \iota^\ell$ where ℓ is a sufficiently large number and $\iota^1 = \iota$ and $\iota^{m+1} = \iota \otimes \iota^m$. Now, assume $2^{(2x+2)^k}_{n} \le |\sigma|_{\max(x,2)}$ where σ is of degree n. Then, we have

$$2^{(2x+2)^k}_{n+1} = 2^{(2^{(2x+2)^k}_{n})} \le 2^{|\sigma|_{\max(x,2)}} \le (|\iota|_{\max(x,2)})^{|\sigma|_{\max(x,2)}} = |\sigma \to \iota|_{\max(x,2)}$$

and $\mathrm{dg}(\sigma \to \iota) = n+1$. ■

THEOREM 9
We have SPACE $2^{\text{LIN}}_{n} \subseteq \mathcal{G}_{2n}$ and TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{G}_{2n+1}$. Furthermore, we have TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{P}_{n+1}$.

PROOF. Let $A \in$ TIME 2^{LIN}_{n+1}, and let $\mathfrak{m}$ be a Turing machine which decides A in time, and thus also in space, $2^{k|x|}_{n+1}$ for some fixed number k. By Lemma 13, there exists a type σ of degree n such that $2^{k|x|}_{n+1} \le |\sigma|_x$. By Lemma 8, we have a T^--term, $\mathrm{T}^{\mathfrak{m}}_{\sigma,\sigma}$ that decides A and a PCF^--term, $\mathrm{PCF}^{\mathfrak{m}}_{\sigma,\sigma}$ that decides A. According to the Lemma, the term $\mathrm{T}^{\mathfrak{m}}_{\sigma,\sigma}$ is of rank $\mathrm{dg}(\sigma)+\mathrm{dg}(\sigma)+1$, i.e. of rank $2n+1$, whereas $\mathrm{PCF}^{\mathfrak{m}}_{\sigma,\sigma}$ is of rank $\max(\mathrm{dg}(\sigma),\mathrm{dg}(\sigma))+1$, i.e. of rank $n+1$. This proves TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{G}_{2n+1}$ and TIME $2^{\text{LIN}}_{n+1} \subseteq \mathcal{P}_{n+1}$ for all $n \in \mathbb{N}$.

Let $A \in$ SPACE 2^{LIN}_{n+1}. Thus, there exists a Turing machine $\mathfrak{m}$ deciding A in space $2^{k|x|}_{n+1}$ and time $2^{k'|x|}_{n+2}$ for some k,k'. By Lemma 13, there exist a type σ of degree n and a type ρ of degree $n+1$ such that $2^{k|x|}_{n+1} \le |\sigma|_x$ and $2^{k'|x|}_{n+2} \le |\rho|_x$. By Lemma 8, we have a T^--term $\mathrm{T}^{\mathfrak{m}}_{\rho,\sigma}$ deciding A. The rank of $\mathrm{T}^{\mathfrak{m}}_{\rho,\sigma}$ is $\mathrm{dg}(\rho)+\mathrm{dg}(\sigma)+1 = 2n+2$. The inclusion SPACE $2^{\text{LIN}}_{0} \subseteq \mathcal{G}_0$ requires a tailored proof, and we omit the details. ■

3.3 Evaluating PCF$^-$-terms by turing machines

LEMMA 14
For every type σ of degree n there exists a polynomial p such that $|\sigma|_x \le 2^{p(x)}_{n}$.

PROOF. We prove the lemma by induction on the structure of σ. The case $\sigma = \iota$ is trivial.

Assume $\sigma=\rho\otimes\tau$. Then $\mathrm{dg}(\sigma)=\max(\mathrm{dg}(\rho),\mathrm{dg}(\tau))$. Hence, we have $\mathrm{dg}(\rho)\leq n$ and $\mathrm{dg}(\tau)\leq n$. The induction hypothesis yields polynomials q and r such that $2_n^{q(x)}\geq|\rho|_x$ and $2_n^{r(x)}\geq|\tau|_x$. The lemma holds since $|\sigma|_x=|\rho|_x\times|\tau|_x\leq 2_n^{q(x)}\times 2_n^{r(x)}\leq 2_n^{r(x)\times q(x)}$.

Assume $\sigma=\rho\to\tau$. Then $\mathrm{dg}(\rho\to\tau)=\max(\mathrm{dg}(\rho)+1,\mathrm{dg}(\tau))$, and hence we have $\mathrm{dg}(\rho)\leq n-1$ and $\mathrm{dg}(\tau)\leq n$. The induction hypothesis yields polynomials q and r such that $2_{n-1}^{q(x)}\geq|\rho|_x$ and $2_n^{r(x)}\geq|\tau|_x$, and the lemma holds since $|\sigma|_x=|\tau|_x^{|\rho|_x}<(2_n^{r(x)})^{2_{n-1}^{q(x)}}=2^{(2_{n-1}^{r(x)}\times 2_{n-1}^{q(x)})}\leq 2_n^{r(x)\times q(x)}$. ■

LEMMA 15
For any type σ of degree $n+1$ there exists a polynomial p such that $\lceil\sigma\rceil_x\leq 2_n^{p(x)}$.

PROOF. First, we note that $\lceil\iota\rceil_x=1$ and $\lceil\rho\otimes\tau\rceil_x=\lceil\rho\rceil_x+\lceil\tau\rceil_x$, and hence

$$\lceil\sigma\rceil_x \text{ is a constant function when } \sigma \text{ is of degree } 0. \qquad (*)$$

We prove the lemma by induction on the structure of σ. Assume $\sigma=\rho\to\tau$ and $\mathrm{dg}(\sigma)\leq n+1$. Then $\mathrm{dg}(\rho)\leq n$ and $\mathrm{dg}(\tau)\leq n+1$. Furthermore, the definition states that $\lceil\rho\to\tau\rceil_x=|\rho|_{x+1}\times\lceil\tau\rceil_x$. By Lemma 14, there exists a polynomial q such that $|\rho|_x\leq 2_n^{q(x)}$. By (*) and the induction hypothesis on τ, the exists a polynomial r such that $\lceil\tau\rceil_x\leq 2_n^{r(x)}$. Hence, there exists a polynomial p such that

$$\lceil\rho\to\tau\rceil_x=|\rho|_{x+1}\times\lceil\tau\rceil_x\leq 2_n^{q(x+1)}\times 2_n^{r(x)}\leq 2_n^{p(x)}.$$

Assume $\sigma=\rho\otimes\tau$ and $\mathrm{dg}(\sigma)\leq n+1$. Then $\mathrm{dg}(\rho)\leq n+1$ and $\mathrm{dg}(\tau)\leq n+1$. By the definition of $\lceil\rho\otimes\tau\rceil_x$ and the induction hypothesis, we have polynomials, p,q,r such that $\lceil\rho\otimes\tau\rceil_x=\lceil\rho\rceil_x+\lceil\tau\rceil_x\leq 2_n^{q(x)}+2_n^{r(x)}\leq 2_n^{p(x)}$. ■

LEMMA 16
Let $M{:}\sigma$ be a PCF^{-}-term of rank $n+1$ satisfying the following requirements

(1) if $F{:}\xi$ is a sub-term of M and $\mathrm{dg}(\xi)>n+1$, then either (i) F is of the form $F\equiv\lambda X^\sigma P^\sigma$ and occurs in the context $\mathrm{Y}_\sigma(F)$ or (ii) F is of the form $F\equiv\lambda x^\iota\lambda X^\sigma P^\sigma$ and occurs in the context $\mathrm{R}_\sigma(G,F,N)$ where $\mathrm{dg}(\sigma)=n+1$
(2) if $\mathrm{Y}_\sigma(F)$ (respectively $\mathrm{R}_\sigma(G,F,N)$) is a sub-term of M and $\mathrm{dg}(\sigma)=n+1$, then F is of the form $F\equiv\lambda X^\sigma P^\sigma$ (respectively $F\equiv\lambda x^\iota\lambda X^\sigma P^\sigma$).

Let $\mathcal{V}$ be any fixed valuation. The value $\mathbf{val}_x^{\mathcal{V}}(M)$ can be computed by a Turing machine running in time $2_{n+1}^{k|x|}$ for some $k\in\mathbb{N}$.

PROOF. We will give an informal algorithm for computing the number $\mathbf{val}_x^{\mathcal{V}}(M)$ where M has the properties stated in the lemma. The algorithm is meant to be carried out by pen and paper, and we will argue that the number of symbols we have to inspect, or write, during the execution is bound by $2_n^{p(x)}$ for some polynomial p. It is too easy to see that the informal algorithm can be implemented by a Turing machine $\mathfrak{m}$ running in time $2_n^{p_0(x)}$ for some polynomial p_0, and hence, there exists $k\in\mathbb{N}$ such that $\mathfrak{m}$ runs in time $2_{n+1}^{k|x|}$.

Let $y_1,\ldots,y_\ell$ be an enumeration of the (bound and unbound) variables occurring in M. The algorithm keeps track of the values assigned to the variables in a list $y_1/a_1,\ldots,y_\ell/a_\ell$, where the natural number a_i is the value currently assigned to the variable y_i. The number ℓ is fixed (rename variables to avoid name conflicts), and we have $a_i<|\sigma|_x$ if a_i is assigned to a variable of type σ. We will have $\mathrm{dg}(\tau)\leq n+1$ for any variable y_i^τ in the list, and by Lemma 14 there exists a polynomial

p_0 such that $2_{n+1}^{p_0(x)}$ bounds every a_i in the list. Thus, each a_i can be represented by a bit string of length $2_n^{p_0(x)}$. This entails that there exists a polynomial $p(x)$ such that the algorithm can assign values to variables, and retrieve values assigned to variables, in time $2_n^{p(x)}$.

The algorithm computes the value $\mathbf{val}_x^{\mathcal{V}}(M)$ by working recursively over the structure of M. Note that the structure of M does not depend on the input x. We will now sketch how the algorithm works in the different cases, and for each case we will argue that the algorithm completes its task within the required time restriction.

Case $M \equiv k_m$. We have $\mathbf{val}_x^{\mathcal{V}}(M) = m \pmod{x+1}$, and the algorithm will simply output the number $m \pmod{x+1}$. The number of steps required to complete this task is obviously bound by $2_n^{p(x)}$ for some polynomial p.

Case $M \equiv y_i$. We have $\mathbf{val}_x^{\mathcal{V}}(M) = \mathcal{V}(y_i)$, and the algorithm will output the number a_i which is stored as a bit string in the assignment list. We have argued above that the number of steps needed to retrieve this number is bounded by $2_n^{p(x)}$ for some polynomial p.

Case $M^\sigma \equiv \lambda z^\pi N^\tau$. We have $\mathbf{val}_b^{\mathcal{V}}(\lambda z^\pi N^\tau) = \sum_{i<|\pi|_b} \mathbf{val}_b^{\mathcal{V}_i^z}(N) \times |\tau|_b^i$, and the algorithm is given by the following informal imperative program.

```
sum:=0;
for i=0,...,|π|_x−1 do {assign i to z; sum:= sum×|τ|_x+val_x^V(N)};
output sum
```

We have $\mathrm{dg}(\pi) \leq n$ since $\mathrm{dg}(\sigma) = \max(\mathrm{dg}(\pi)+1, \mathrm{dg}(\tau)) \leq n+1$. By Lemma 14, there exist polynomials p_0 and p_1 such that $|\pi|_x \leq 2_n^{p_0(x)}$ and $|\sigma|_x \leq 2_{n+1}^{p_1(x)}$. Hence, the loop will be executed no more than $2_n^{p_0(x)}$ times. Furthermore, the number computed into the register **sum** is less than $|\sigma|_x$ and thus bounded by $2_{n+1}^{p_1(x)}$. The number of bits required to represent the number is bounded by $2_n^{p_1(x)}$. The induction hypothesis yields a polynomial p_2 such that the number of steps required to compute $\mathbf{val}_x^{\mathcal{V}}(N)$ is bounded by $2_n^{p_2(x)}$. It follows that there exists a polynomial p such that the number of steps required to compute $\mathbf{val}_x^{\mathcal{V}}(M)$ is bounded by $2_n^{p(x)}$.

Case $M^\sigma \equiv \mathrm{Y}_\sigma(N)$. We have $\mathbf{val}_x^{\mathcal{V}}(\mathrm{Y}_\sigma N) = f^{\lceil\sigma\rceil_x}(0)$ where $f(y) = \mathbf{val}_x^{\mathcal{V}}(N)[y]_x$. The proof splits into the two cases: (i) $\mathrm{dg}(\sigma) = n+1$ and (ii) $\mathrm{dg}(\sigma) \leq n$. We will give the proof for case (i). Case (ii) is easier, and we leave the proof to the reader.

When $\mathrm{dg}(\sigma) = n+1$, the second requirement in the lemma states that N is of the form $N \equiv \lambda z^\sigma P^\sigma$. The algorithm for computing $\mathbf{val}_x^{\mathcal{V}}(\mathrm{Y}_\sigma \lambda z P)$ is given by the following imperative program.

```
a:=0; for i=0,...,⌈σ⌉_x do {assign a to z; a:=val_x^V(P)}; output a
```

By Lemma 15, there exists a polynomial q such that $\lceil\sigma\rceil_x \leq 2_n^{q(x)}$, and thus, the loop will be executed no more than $2_n^{q(x)}$ times. By the induction hypothesis, the computations of the value $\mathbf{val}_x^{\mathcal{V}}(P)$ requires no more than $2_n^{r(x)}$ steps for some polynomial r. This entails that the entire program require no more than $2_n^{p(x)}$ steps for some polynomial p.

Case $M^\sigma \equiv (N^{\rho\to\sigma} P^\rho)$. We have $\mathbf{val}_x^{\mathcal{V}}((NP)) = \mathbf{val}_x^{\mathcal{V}}(N)[\mathbf{val}_x^{\mathcal{V}}(P)]_x$. By the first requirement in the lemma, the degrees of $\rho \to \sigma$ and ρ are less or equal to $n+1$. By the induction hypothesis, there exist polynomials q and r such that the values of $\mathbf{val}_x^{\mathcal{V}}(N)$ and $\mathbf{val}_x^{\mathcal{V}}(N)$ can be computed in, respectively, $< 2_n^{q(x)}$ and $< 2_n^{r(x)}$ steps. It follows that the value of $\mathbf{val}_x^{\mathcal{V}}((NP))$ can be computed within the required time constraints.

The case when $M^\sigma \equiv \mathrm{R}_\sigma(G,F,N)$ is similar to the case when $M^\sigma \equiv \mathrm{Y}_\sigma(N)$. The remaining cases, i.e. $M \equiv \mathbf{fst} M_1$, $M \equiv \mathbf{snd}\ M'$ and $M \equiv \langle M_1, M_2 \rangle$, are fairly straightforward, and we omit the details. ∎

THEOREM 10
We have $\mathcal{P}_{n+1} \subseteq \text{TIME } 2^{\text{LIN}}_{n+1}$.

PROOF. Assume $A \in \mathcal{P}_{n+1}$. The definition of $\mathcal{P}_{n+1}$ says that there exists a closed PCF$^-$-term $M^{\iota\to\iota}$ of rank $n+1$ such that $Mk_x \overset{\star}{\rhd} k_0$ if $x \in A$, and $Mk_x \overset{\star}{\rhd} k_1$ if $x \notin A$. By applying the reduction rules of the typed λ-calculus, i.e. α-conversions and β-conversions, the term M (of rank $n+1$) can be converted into a term N (also of rank $n+1$) satisfying the two requirements of Lemma 16. Let $f_A(x) = \mathbf{val}_{\max(x,m)+1}(Nk_x)$ where m is the greatest m such that k_m occurs in M. By our Soundness Theorem (7), we have $f_A(x) = 0$ if $x \in A$, and $f_A(x) = 1$ if $x \notin A$. Lemma 16 says that $f_A(x)$ can be computed by a Turing machine running in time $2^{k|\max(x,m)+1|}_{n+1}$ for some $k \in \mathbb{N}$, and thus, in time $2^{k'|x|}_{n+1}$ for some $k' \in \mathbb{N}$. Hence, we have $A \in \text{TIME } 2^{\text{LIN}}_{n+1}$ and the inclusion $\mathcal{P}_{n+1} \subseteq \text{TIME } 2^{\text{LIN}}_{n+1}$ holds. ∎

References

[1] A. Asperti and H. G. Mairson. Parallel beta reduction is not elementary recursive. *Information and Computation*, **170**, 49–80, 2001.

[2] J. Avigad and S. Feferman. Gödel's functional interpretation. In *Handbook of Proof Theory*. S. Buss, ed., Elsevier, 1998.

[3] G. M. Barra, L. Kristiansen, and P. J. Voda. Nondeterminism without Turing Machines. In *Computation and Logic in the Real World CiE 2007, Siena, Italy, June 18-23, 2007*, S. B. Cooper *et al*., eds. Local Proceedings, Technical report no. 487, June 2007, Dipartimento di Scienze Matematiche e Informatiche R. Magari, University of Sienna.

[4] A. Beckmann and A. Weiermann. Characterizing the elementary recursive functions by a fragment of Gödel's T. *Archive for Mathematical Logic*, **39**, 475–491, 2000.

[5] S. J. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity*, **2**, 97–110, 1992.

[6] S. Bellantoni, K.-H. Niggl, and H. Schwichtenberg. Ramification, modality and linearity in higher type recursion. *Annals of Pure and Applied Logic*, **104**, 17–30, 2000.

[7] A. Bel'tyukov. A machine description and the hierarchy of initial Grzegorczyk classes. *Journal of Soviet Mathematics*, 1982. *Zap. Naucn. Sem. Leninigrad. Otdel. May. Inst. Steklov. (LOMI)*, **88**, 30–46, 1979.

[8] P. Clote. Computation models and function algebra. In *Handbook of Computability Theory*. E. Griffor, ed., pp. 589–681. Elsevier, 1999.

[9] M. A. Esbelin and M. More. Rudimentary relations and primitive recursion: a toolbox. *Theoretical Computer Science*, **193**, 129–148, 1998.

[10] R. Gandy. Some relations between classes of low computational complexity. *Bulletin of London Mathematical Society*, **16**, 127–134, 1984.

[11] J.-Y. Girard. Light linear logic. *Information Computation*, **14**, 175–204, 1998.

[12] J.-Y. Girard, A. Scedrov, and P. J. Scott. Bounded linear logic: a modular approach to polynomial time computability. *Theoretical Computer Science*, **97**, 1–66, 1992.

[13] A. Goerdt. Characterizing complexity classes by higher type primitive recursive definitions. *Theoretical Computer Science*, **100**, 45–66, 1992.

[14] A. Goerdt and H. Seidl. Characterizing complexity classes by higher type primitive recursive definitions, part ii. In *Aspects and Prospects of Theoretical Computer Science*, J. Dassow and J. Kelemen, eds, Vol. 464 of *Lecture Notes in Computer Science*, pp. 148–158, Springer, 1990.

[15] A. Grzegorczyk. *Some Classes of Recursive Functions. Rozprawy Matematyczne*, No. IV, Warszawa, 1953.

[16] Y. Gurevich. Toward logic tailored for computational complexity. In *Computation and Proof Theory (Aachen, 1983)*, M. M. Richter *et al*., eds, Vol. 1104 of *Lecture Notes in Mathematics*, pp. 175–216. Springer, 1984.

[17] N. Immerman. Descriptive complexity. *Graduate Texts in Computer Science*. Springer-Verlag, 1999.

[18] N. Jones. The expressive power of higher-order types or, life without CONS. *Journal of Functional Programming*, **11**, 55–94, 2001.

[19] L. Kristiansen. Complexity-theoretic hierarchies induced by fragments of Gödel's T. *Theory of Computing Systems*, **43**, 516–541, 2008.

[20] L. Kristiansen and G. Barra. The small Grzegorczyk classes and the typed λ-calculus. In *CiE 2005: New Computational Paradigms*. S. B. Cooper *et al*., eds, Vol. 3526 of *Lecture Notes in Computer Science*, pp. 252–262. Springer, 2005.

[21] L. Kristiansen and P. Voda. Programming languages capturing complexity classes. *Nordic Journal of Computing*, **12**, 1–27, 2005.

[22] L. Kristiansen and P. Voda. The trade-off theorem and fragments of Gödel's T. In *TAMC 2006: Theory and Applications of Models of Computation*. J.-Y. Cai *et al.*, eds, Vol. 3959 of *Lecture Notes in Computer Science*, pp. 654–674. Springer, 2006.

[23] M. Kutylowski. Small Grzegorczyk classes. *Journal of the London Mathematical Society*, **36**, 193–210, 1987.

[24] H. G. Mairson. A simple proof of a theorem of Statman. *Theoretical Computer Science*, **103**, 387–394, 1992.

[25] P. Odifreddi. *Classical Recursion Theory II. Vol. 143 of Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., 1999.

[26] J. Paris and A. Wilkie. Counting problems in bounded arithmetic. In *Methods in Mathematical Logic*. Vol. 1130 of *Lecture Notes in Mathematics*, pp. 317–340. Springer, 1985; Proceedings, Caracas, 1983.

[27] H. Rose. *Subrecursion. Functions and Hierarchies*. Clarendon Press, 1984.

[28] H. Schwichtenberg. Classifying recursive functions. In *Handbook of Computability Theory*. E. Griffor, ed., Elsevier, pp. 533–586, 1996.

[29] H. Simmons. The realm of primitive recursion. *Archive for Mathematical Logic*, **27**, 177–188, 1988.

[30] H. Simmons. *Derivation and Computation. Taking the Curry-Howard Correspondence Seriously. Cambridge Tracts in Theoretical Computer Science. 51*. Cambridge University Press, 2000.

[31] T. Streicher. *Domain-Theoretic Foundations of Functional Programming*. World Scientific Publishing Co. Pte. Ltd., 2006.

Received 14 November 2008