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 Section 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 $\lambda$-calculus and Gödel’s $T$. For more on Gödel’s $T$, see Avigad & Feferman [2].

Definition. We define the types recursively:

- $\iota$ is a type (primitive type);
- $\sigma \otimes \tau$ is a type if $\sigma$ and $\tau$ are types (product types)
- $\sigma \rightarrow \tau$ is a type if $\sigma$ and $\tau$ are types (arrow types).

The notation $\sigma_1, \sigma_2, \ldots, \sigma_n \rightarrow \tau$ is shorthand for $\sigma_1 \rightarrow (\sigma_2 \rightarrow (\ldots (\sigma_n \rightarrow \tau) \ldots))$, and $\text{typ}$ denotes the set of all types.

We define the terms of the typed $\lambda$-calculus:

- We have an infinite supply of variables $x^\sigma_0, x^\sigma_1, x^\sigma_2, \ldots$ for each type $\sigma$. A variable of type $\sigma$ is a term of type $\sigma$
- $\lambda x M$ is a term of type $\sigma \rightarrow \tau$ if $x$ is a variable of type $\sigma$ and $M$ is a term of type $\tau$ ($\lambda$-abstraction)
- $(MN)$ is a term of type $\tau$ if $M$ is a term of type $\sigma \rightarrow \tau$ and $N$ is a term of type $\sigma$ (application)
- $(M, N)$ is a term of type $\sigma \otimes \tau$ if $M$ is a term of type $\sigma$ and $N$ is a term of type $\tau$ (pairing)
- $\text{fst} M, \text{snd} M$ is a term of type $\sigma (\tau)$ if $M$ is a term of type $\sigma \otimes \tau$ (projections).
Next we define the reduction rules of the typed \(\lambda\)-calculus. We have the following \(\beta\)-conversions: \((\lambda x M)N \triangleright M[x := N]\) if \(x \notin \text{FV}(N)\); \(\text{fst}(M, N) \triangleright M\); and \(\text{snd}(M, N) \triangleright N\). Further, we have \(\alpha\)-conversion, \(\eta\)-conversion and all the other standard reduction rules, i.e., \((MN) \triangleright (MN')\) if \(N \triangleright N'\); \((MN) \triangleright (M'N)\) if \(M \triangleright M'\); etcetera.

The calculus \(T^-\) is the typed \(\lambda\)-calculus extended with

- for each \(i \in \mathbb{N}\) a constant \(k_i\) of type \(\iota\)
- recursor terms \(R_\sigma(G, F, F^{\iota \rightarrow \sigma}, N)\) of type \(\sigma\), that is, \(R_\sigma(G, F, N)\) is a term of type \(\sigma\) if \(G\) and \(F\) and \(N\) are terms of, respectively, type \(\sigma\), type \(\iota\), \(\sigma \rightarrow \sigma\) and type \(\iota\)
- reduction rules of the form

\[
R_\sigma(G, F, k_0) \triangleright G \quad \text{and} \quad R_\sigma(G, F, k_{n+1}) \triangleright F(k_n, R_\sigma(G, F, k_n))
\]

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 \(M^\sigma\) and \(M: \sigma\) are two alternative ways to signify that the term \(M\) is of type \(\sigma\); \(M(M_1, M_2)\) denotes the term \(((MM_1)M_2)\), etcetera. If we write \(M^k\) where \(M\) is a term and \(k\) is a natural number, then \(M^k\) denotes the term \(\underbrace{M \cdots M}_{k\text{ times}}\) in a row, e.g., \(M^3N\) denotes the term \(MMN\). We will use \(\triangleright\) to denote the transitive-reflexive closure of the reducibility relation \(\triangleright\); and \(=\) to denote the symmetric-transitive-reflexive closure of \(\triangleright\); and \(\equiv\) to denote syntactical equality between terms.

As the definition above shows, \(T^-\) is more or less a programming language version of Godel’s \(T\) where the successor function \(S: \iota \rightarrow \iota\) 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 given above.

The programming language \(T\) is \(T^-\) extended with the successor function \(S: \iota \rightarrow \iota\) and reduction rules of the form \(S(k_i) \triangleright k_{i+1}\). It is well known that any closed \(T\)-term of type \(\iota\), and a fortiori any \(T^-\)-term of type \(\iota\), normalises to a unique constant \(k_i\). Thus, a closed term \(M: \iota \rightarrow \iota\) defines a function \(f: \mathbb{N} \rightarrow \mathbb{N}\), and the value \(f(n)\) can be computed by normalising 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 the next definition we stratify the problems decidable in \(T^-\) into a hierarchy.

**Definition.** A **problem** is a subset of \(\mathbb{N}\). A term \(M: \iota \rightarrow \iota\) **decides** a problem \(A\) when

\[
M(k_0) \triangleright \begin{cases} \; k_0 & \text{if } x \in A \\ \; k_1 & \text{otherwise} \end{cases}
\]

We define the **degree of the type \(\sigma\)**, written \(\text{dg}(\sigma)\), by recursion on the structure of the type \(\sigma\):
The rank of a term $M$, written $\text{Rk}(M)$, is the greatest number $n$ such that $n \leq \text{dg}(\sigma)$ for any recursor term $R_{\sigma}(\ldots)$ occurring in $M$.

Various schemes have been 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. Simmons [35] and Beckmann & Weiermann [4]. 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 Odifreddi [30].

**Definition.** 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 $2^x_0 = x$ and $2^x_{i+1} = 2^{2^x_i}$. For $i \in \mathbb{N}$, we define $\text{TIME } 2^i_0$ (SPACE $2^i_0$) to be the set of problems decidable by a deterministic Turing machine working in time (space) $2^{k|x|}$ for some fixed $k \in \mathbb{N}$.

It is trivial that $\text{TIME } 2^i_0 \subseteq \text{SPACE } 2^i_0$ and $\text{SPACE } 2^i_0 \subseteq \text{TIME } 2^{i+1}_0$, and thus, we have an alternating space-time hierarchy

$\text{SPACE } 2^0_0 \subseteq \text{TIME } 2^1_0 \subseteq \text{SPACE } 2^1_0 \subseteq \text{TIME } 2^2_0 \subseteq \text{SPACE } 2^2_0 \subseteq \text{TIME } 2^3_0 \subseteq \ldots$

The three classes at the bottom of the hierarchy are often called respectively LSPACE, EXP, and EXPSPACE in the literature. It is well known, and quite obvious, that we have $\text{SPACE } 2^i_0 \subseteq \text{SPACE } 2^i_{i+1}$ and $\text{TIME } 2^i_0 \subseteq \text{TIME } 2^i_{i+1}$ for any $i \in \mathbb{N}$. Thus, we know that at least one of the two inclusions

$\text{SPACE } 2^i_0 \subseteq \text{TIME } 2^i_{i+1} \subseteq \text{SPACE } 2^i_{i+1}$

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

$\text{TIME } 2^i_0 \subseteq \text{SPACE } 2^i_0 \subseteq \text{TIME } 2^i_{i+1}$
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 \( \mathcal{G}_{0} \subseteq \mathcal{G}_{1} \subseteq \mathcal{G}_{2} \subseteq \ldots \) 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 (Kristiansen & Voda [26][27]).** We have \( \text{SPACE} \ 2^{n_{\alpha}} = \mathcal{G}_{2n} \) and \( \text{TIME} \ 2^{n_{\alpha}} = \mathcal{G}_{2n+1} \).

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

\[
\text{LOGSPACE} \subseteq \text{TIME} \ 2^{n_{\alpha}} \subseteq \text{SPACE} \ 2^{n_{\alpha}} \subseteq \text{TIME} \ 2^{n_{\alpha}} \subseteq \text{SPACE} \ 2^{n_{\alpha}} \subseteq \ldots
\]

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 \( \mathcal{C}_{i} \subseteq \mathcal{C}_{i+2} \), 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 \( \text{TIME} \ 2^{n_{\alpha}} \) and \( \text{SPACE} \ 2^{n_{\alpha}} \) are the classes usually denoted respectively \( \text{P} \) and \( \text{PSPACE} \) in the literature, so the notorious open problem \( \text{LOGSPACE} \nsubseteq \text{P} \nsubseteq \text{PSPACE} \) 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 \( \text{SPACE} \ 2^{n_{\alpha}} \) and \( \text{TIME} \ 2^{n_{\alpha}} \) is that the two classes cannot be equal. So, it is known that e.g. \( \text{LINSPACE} \neq \text{P} \), but it is an open problem if \( \text{LINSPACE} \) is strictly included in \( \text{P} \), or if \( \text{P} \) is strictly included in \( \text{LINSPACE} \), or if neither of the two classes is included in the other.

\( T^{-} \) can easily be modified into a programming language doing recursion on bitstrings in place of recursion on natural numbers. We throw away all the constants \( k_{0}, k_{1}, k_{2}, \ldots \) and introduce a constant \( k_{\alpha} \) of type \( \alpha \) for each bit string \( \alpha \). The recursion terms needs to be adjusted accordingly, that is, we need recursion terms of the form \( R_{\alpha}(G^{\alpha}, F_{0}^{\alpha}, \sigma_{\alpha}, F_{1}^{\alpha}, \sigma_{\alpha}, N^{\alpha}) \) and the reduction rules

- \( R_{\varepsilon}(G, F_{0}, F_{1}, k_{\alpha}) \Rightarrow G \) where \( \varepsilon \) denotes the empty string
- \( R_{\alpha}(G, F_{0}, F_{1}, k_{\alpha}) \Rightarrow F_{b}(k_{\alpha}, R_{\alpha}(G, F_{0}, F_{1}, k_{\alpha})) \) where \( b \in \{0, 1\} \) and \( \alpha \in \{0, 1\}^{*} \).

This modified version of \( T^{-} \) induces a hierarchy \( \mathcal{G}^{b} = \bigcup_{n \in \mathbb{N}} \mathcal{G}_{n}^{b} \) which is defined analogously to the hierarchy \( \mathcal{G} \).
Theorem 2 (Kristiansen & Voda [26][27]). We have \( \text{LOGSPACE} = \mathcal{G}_0^b \) and

\[
\text{SPACE } 2^i = \mathcal{G}_1^b \quad \text{and} \quad \text{TIME } 2^{i+1} = \mathcal{G}_2^b
\]

Theorem 1 and Theorem 2 are proved for the first time in Kristiansen & Voda [26]. The proofs in [26] of the inclusions

\[
\mathcal{G}_2^n \subseteq \text{SPACE } 2^n \quad \text{and} \quad \mathcal{G}_2^{n+1} \subseteq \text{TIME } 2^{n+1}
\]

are inspired by Goerdt and Seidel's work in finite model theory. See Goerdt & Seidl [13] and Goerdt [14]. Essentially different proofs of these inclusions can be found in Kristiansen & Voda [27]. The proofs in [27] are based on an adaption of Schwichtenberg’s Trade-off Theorem to a complexity-theoretic context. For more on Schwichtenberg’s Theorem see [33].

1.3 \( T^- \) and Subrecursive Classes

A severely refined version of the \( T^- \)-hierarchy \( \mathcal{G} \) appears in Kristiansen [22]. In [22], we work with sum types in addition to product and arrow types and with a basic type \( q \) in addition to the basic type \( \iota \). The type \( q \) is a unit type that contains only one sole element. We define a well ordering \( \preceq \) of the types, and then, for each type \( \sigma \), 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 \( \mathcal{G} = \bigcup_{\sigma \in \text{TYP}} \mathcal{G}_\sigma \). 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 \( E_0, E_1 \) and \( E_2 \) are typical examples. The class \( E_0 \) is the closure of \( \mathcal{u}_n \) (projections), \( S \) (successor) \( 0 \) (zero) under composition and bounded primitive recursion; \( E_1 \) and \( E_2 \) are defined similarly, but the sets of initial functions are extended with respectively \( + \) (addition) and \( \times \) (multiplication). Now, let \( E_0^*, E_1^* \) and \( E_2^* \) respectively denote the problems (i.e the 0-1-valued functions) of the Grzegorczyk classes \( E_0, E_1 \) and \( E_2 \), and let \( \Delta_0^N \) be the set problems definable in Peano Arithmetic by a \( \Delta_0^0 \) statement.

Theorem 3 (Kristiansen [22]). We have

- \( \Delta_0^N \subseteq \mathcal{G}_@q \)
- \( \mathcal{G}_@ = E_0^* \)
- \( \mathcal{G}_@ = E_1^* \)
- \( \mathcal{G}_@ = E_2^* \)

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 \( \Delta_0^N \subseteq E_0^* \subseteq E_1^* \subseteq E_2^* \), but it is not known whether any of the inclusions are strict, indeed it is open if the inclusion \( \Delta_0^N \subseteq E_2^* \) is strict. It is proved in Bel’tyukov [7] that \( E_1^* = E_2^* \) implies \( E_0^* = E_2^* \). Furthermore, we know that \( \Delta_0^N = E_0^* \) implies \( \Delta_0^N = E_2^* \) (see Kristiansen & Barra [23]). 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 Clote [8], Rose [32], Kutyłowski [28], Esbelin & More [9], Gandy [10], Paris & Wilkie [31], Kristiansen & Barra [23].

We expect a lot of more or less natural subrecursive classes to match classes in the refined $G$-hierarchy, e.g. the hierarchy of subrecursive classes studied in Kristiansen & Barra [23].

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, that is, 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

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

and the two accompanying reduction rules $(M|N) \triangleright M$ and $(M|N) \triangleright N$. This extension will indeed yield a non-deterministic version of any class in our refined hierarchy, also those classes that we cannot characterise 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 classes relate 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 Barra, Kristiansen & Voda [3]. In the current paper we will introduce general recursion into the picture by extending $T^-$ with fixed-point terms.

Definition. The calculus PCF$^-$ is the calculus $T^-$ extended with with a fixed-point term $Y_\sigma M$ of type $\sigma$ for each term $M$ of type $\sigma \rightarrow \sigma$ and reduction rules of the form $Y_\sigma M \triangleright M(Y_\sigma M)$.

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

We define the hierarchy 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. Streicher [36] 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, that is, PCF without recursor terms, but with boolean types, predecessor terms, etcetera.

1.5 Main Results and Related Research

The main result of this paper is the following theorem.

**Theorem 4 (Main).** We have $\text{time } 2^{\text{lin } n+1} = \mathcal{P}_{n+1}$.

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

$$\text{space } 2^{\text{lin } n} \subseteq \mathcal{G}_2n \quad \text{time } 2^{\text{lin } n+1} \subseteq \mathcal{G}_{2n+1} \quad \text{time } 2^{\text{lin } n+1} \subseteq \mathcal{P}_{n+1}.$$  

These proofs, which are given in Section 3.1 and Section 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 $\text{space } 2^{\text{lin } n} \subseteq \mathcal{G}_2n$ and $\text{time } 2^{\text{lin } n+1} \subseteq \mathcal{G}_{2n+1}$ are more transparent and direct than those given elsewhere.

The proof of the inclusion $\mathcal{P}_{n+1} \subseteq \text{time } 2^{\text{lin } n+1}$ 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 $\mathcal{P}_{n+1} \subseteq \text{time } 2^{\text{lin } n+1}$ 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. Simmons [34], Bellanoni & Cook [5]; work on finite model theory by e.g. Gurevich [16], Immerman [17], Goerdt [14]; work on linear logic by e.g. Girard [11], Girard, Scedrov & Scott [12]; work on proof theory by e.g. Bellantoni, Niggl & Schwichtenberg [6]; work on the typed $\lambda$-calculus; and so on. This is research with different starting points, different emphasises and different motivations. In the end, some of this research might turn out to be more similar than expected, and the author realises that he should be careful to claim too much originality for some of the research presented in this paper.

Work closely related to ours includes work by Mairson, not know by the author until recently. The techniques used in Mairson [29] and Asperti & Mairson
to simulate Turing machines in the typed \( \lambda \)-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 characterisation similar to ours of the hierarchy \( \bigcup_{n \in \mathbb{N}} \text{TIME } 2^{2n} \). 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 capture a slightly different hierarchy than ours, that is, \( \bigcup_{n \in \mathbb{N}} \text{TIME } 2^{2n} \) in contrast to \( \bigcup_{n \in \mathbb{N}} \text{TIME } 2^{\text{lin} n} \).

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 paper 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 paper are probably the domain theory developed in Section 2 and the application of this theory in Section 3.

2 Interpreations of \( PCF^- \)-terms

We assume some experience with domain theory. For more on the subject, see e.g. Streicher [36].

2.1 Interpreations in Finite Domains

Definition. For any natural number \( b > 1 \) we define the finite domain \( D_b^\sigma \), and the binary relation \( \sqsubseteq_\sigma \) over \( D_b^\sigma \), recursively over the structure of the type \( \sigma \). Let \( D_b=\{\bot,0,1,\ldots,b-1\} \) and

\[
d \sqsubseteq_\sigma e \iff d = \bot \lor d = e .
\]

Let \( D_{\rho \rightarrow \tau}^b = \{ f : D_{\rho}^b \rightarrow D_{\tau}^b \mid d \sqsubseteq_\rho e \Rightarrow f(d) \sqsubseteq_\tau f(e) \} \) and

\[
d \sqsubseteq_{\rho \rightarrow \tau} e \iff \forall a \in D_{\rho}^b \; [d(a) \sqsubseteq_\tau e(a)] .
\]

Let \( D_{\rho \otimes \tau}^b = D_{\rho}^b \times D_{\tau}^b \) and

\[
d \sqsubseteq_{\rho \otimes \tau} e \iff \text{fst}(d) \sqsubseteq_\rho \text{fst}(e) \land \text{snd}(d) \sqsubseteq_\tau \text{snd}(e)
\]

where \( \text{fst}(x) \) and \( \text{snd}(x) \) denote respectively the first and the second component of the pair \( x \in D_{\rho}^b \times D_{\tau}^b \). We use \( \bot_\sigma \) to denote the unique element in \( D_0^\sigma \) such that \( \bot_\sigma \sqsubseteq_\sigma d \) holds for any \( d \in D_0^\sigma \). Occasionally, we will suppress the subscript and just write \( \bot \).
We define the cardinality of the type $\sigma$ at base $b$, written $|\sigma|_b$, by recursion on the structure of the type $\sigma$: $|i|_b = b$; $|\rho \otimes \tau|_b = |\rho|_b \times |\tau|_b$; $|\rho \rightarrow \tau|_b = |\rho|_b \times |\tau|_b$.

We define the domain height of the type $\sigma$ at base $b$, written $\sigma|_b$, by recursion on the structure of the type $\sigma$: $[i]_b = 1$; $[\rho \otimes \tau]_b = [\rho]_b + [\tau]_b$; $[\rho \rightarrow \tau]_b = [\rho]_b \times [\tau]_b$. $\Box$

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

**Proof.** We prove the lemma by induction on the structure of $\sigma$. We have $D^b_\emptyset = \{\bot, 0, 1, \ldots, b-1\}$ and $|\emptyset|_{b+1} = b+1$ straightaway from the definitions, and thus the lemma holds when $\sigma = \emptyset$. Assume by induction hypothesis that $|D^b_\rho| \leq |\rho|_{b+1}$ and $|D^b_\tau| \leq |\tau|_{b+1}$. Now, $D^b_{\rho \rightarrow \tau}$ is a subset of the functions from $D^b_\rho$ into $D^b_\tau$, and hence we have $|D^b_{\rho \rightarrow \tau}| \leq |\rho|_{b+1}$. Moreover, $D^b_{\rho \otimes \tau} = D^b_\rho \times D^b_\tau$, and hence we have $|D^b_{\rho \otimes \tau}| \leq |\rho|_{b+1} \times |\tau|_{b+1} = |\rho \otimes \tau|_{b+1}$. $\Box$

**Lemma 2.** Let $d_0, \ldots, d_m$ be elements in the finite domain $D^b_\sigma$ such that $d_i \subseteq_{\sigma} d_{i+1}$ and $d_i \neq d_{i+1}$ for all $i < m$. Then we have $m \leq |\sigma|_b$.

**Proof.** Let $d \subseteq_{\sigma} d'$ denote that $d \subseteq_{\sigma} d'$ and $d \neq d'$. We prove the lemma by induction on the structure of $\sigma$.

We have $d \subseteq_{\sigma} d'$ if and only if $d = \bot$ and $d' \in \mathbb{N}$. Hence, if $d_0 \subseteq_{\sigma} \ldots \subseteq_{\sigma} d_m$, then $m \leq 1$. Our definitions states that $[i]_b = 1$, and thus the lemma holds when $\sigma = \emptyset$.

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

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

where $m > [\rho \otimes \tau] = [\rho]_b + [\tau]_b$. 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 $m > [\rho]_b + [\tau]_b$, (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
   $$a_{i_0} \subseteq_{\rho} \ldots \subseteq_{\rho} a_{i_k} \quad \text{and} \quad k > [\rho]_b.$$  
2. there exists a sub-sequence $b_{i_0}, \ldots, b_{i_\ell}$ of $b_0, \ldots, b_m$ such that
   $$b_{i_0} \subseteq_{\tau} \ldots \subseteq_{\tau} b_{i_\ell} \quad \text{and} \quad \ell > [\tau]_b.$$  

The former case contradicts our induction hypothesis on $\rho$, the latter case contradicts our induction hypothesis on $\tau$.

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

$$d_0 \subseteq_{\rho \rightarrow \tau} \ldots \subseteq_{\rho \rightarrow \tau} d_m$$

where $m > [\rho \rightarrow \tau] = [\rho]_{b+1} \times [\tau]_b$. 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
Lemma 4. We omit the details. The proof is fairly straightforward. Use induction over the structure of $M$. If $\sigma \in \text{TYP} D^b_\sigma$ such that $\mathcal{A}(x^\sigma) \in D^b_\sigma$, that is, $\mathcal{A}$ assigns values in the finite domain $D^b_\sigma$ to variables of type $\sigma$. We define the assignment $\mathcal{A}^+_i$ by $\mathcal{A}^+_i(y) = \mathcal{A}(y)$ for any variable $y$ different from $x$. A PCF$^-$-term $M$ belongs to the fragment PCF$^-$ if $i < b$ for each constant $k_i$ occurring in $M$.

We define the (base $b$) interpretation of the PCF$^-$-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 $k\ell$
2. $[x]_{\mathcal{A}} = \mathcal{A}(x)$ for any variable $x$
3. $[(M N)]_{\mathcal{A}} = [M]_{\mathcal{A}} [N]_{\mathcal{A}}$
4. $[\lambda x. M]_{\mathcal{A}} = f$ where the function $f$ is defined by $f(d) = [M]_{\mathcal{A}^+_i}$
5. $[\text{fst} M]_{\mathcal{A}} = \text{fst}([M]_{\mathcal{A}})$
6. $[\text{snd} M]_{\mathcal{A}} = \text{snd}([M]_{\mathcal{A}})$
7. $[[M_1, M_2]]_{\mathcal{A}} = ([M_1]_{\mathcal{A}}, [M_2]_{\mathcal{A}})$
8. $[\text{Re}_\sigma(G, F, N)]_{\mathcal{A}} = \begin{cases} \bot_{\sigma} & \text{if } [N]_{\mathcal{A}} = \bot, \\ f(n) & \text{otherwise} \end{cases}$

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. $[[Y_\sigma M]]_{\mathcal{A}} = f^k(\bot_{\sigma})$ where $k = [\sigma]_b$ 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 $[M]_{\mathcal{A}}$ when $M$ is closed. □

Lemma 3. If $M$ is a PCF$^-$-term of type $\sigma$ 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$^-$-term, and let $\mathcal{A}$ be a base $b$ assignment. If $M \triangleright N$, then $[M]_{\mathcal{A}} = [N]_{\mathcal{A}}$.

Proof. The interesting case is the reduction rule $Y_\sigma M \triangleright M(Y_\sigma M)$. Let $f = [M]_{\mathcal{A}}$. We have $f \in D^b_{\sigma \to \sigma}$, and thus $f^k(\bot) \subseteq f^{k+1}(\bot)$ for any $k \in \mathbb{N}$. By Lemma 2, we have $f^k(\bot) = f^{k+1}(\bot)$ for any $k \geq [\sigma]_b$. Hence $[[Y_\sigma M]]_{\mathcal{A}} = f^{[\sigma]_b}(\bot) = f^f^{[\sigma]_b}(\bot) = [M]_{\mathcal{A}}[[Y_\sigma M]]_{\mathcal{A}} = [M(Y_\sigma M)]_{\mathcal{A}}$.

The proofs for the remaining reduction rules are standard, and we omit the details. □
Theorem 5 (Soundness I). For any closed PCF\textsuperscript{b}\textsuperscript{−}-term \( M \) of type \( \iota \), we have \( [M] = n \) whenever \( M \triangleright k_n \).

Proof. Assume \( M \triangleright k_n \), i.e., we have a reduction sequence
\[
M \equiv M_0 \triangleright M_1 \triangleright \ldots \triangleright M_\ell \equiv k_n .
\]
By Lemma 4, we have \( [M_i] = [M_{i+1}] \) for any \( i < \ell \). Thus, we have \( [M] = n \) since \( [k_n] = n \). \( \square \)

2.2 Interpretations in Finite Total Function Spaces

We will now define the interpretation \( \llbracket M \rrbracket_A \) interpreting any PCF\textsuperscript{b}\textsuperscript{−}-term \( M \) as a finite total object.

Definition. For any natural number \( b > 1 \) we define the \textit{finite total objects} \( T^b_{\sigma} \), and the binary relation \( d \preceq_\sigma t \) where \( d \in D^b_{\sigma} \) and \( t \in T^b_{\sigma} \), recursively over the structure of the type \( \sigma \). Let \( T^b_\iota = \{0, \ldots, b-1\} \), and let
\[
d \preceq_\iota t \iff d = \bot \lor d = t .
\]
Let \( T^b_{\rho \rightarrow \tau} \) be the set of all total function from \( T^b_\rho \) into \( T^b_\tau \), and let
\[
d \preceq_{\rho \rightarrow \tau} t \iff d(i) \preceq_\tau t(j) \text{ for all } i \preceq_\rho j .
\]
Let \( T^b_{\rho \otimes \tau} = T^b_\rho \times T^b_\tau \), and let
\[
a \preceq_{\rho \otimes \tau} b \iff \text{fst}(a) \preceq_\rho \text{fst}(a) \land \text{snd}(b) \preceq_\tau \text{snd}(b) .
\]

Let \( A \) be a base \( b \) assignment. An \( A\text{-compatible total assignment} \( \hat{A} \) is a total function from the set of PCF\textsuperscript{−}-variables into the set \( \bigcup_{\sigma \in \text{TY} \iota} T^b_{\sigma} \) such that \( \hat{A}(x^\sigma) \in T^b_{\sigma} \) and \( A(x) \preceq_\sigma \hat{A}(x) \).

We define the \textit{total (base \( b \)) interpretation} of the PCF\textsuperscript{b}\textsuperscript{−} term \( M \) under the \( A\text{-compatible assignment} \( \hat{A} \), written \( [M]_{\hat{A}} \), by recursion over the structure of \( M \). Let

1. \( [k]_{\hat{A}} = \ell \) for any constant \( k_\ell \)
2. \( [x]_{\hat{A}} = \hat{A}(x) \) for any variable \( x \)
3. \( [MN]_{\hat{A}} = [M]_{\hat{A}} [N]_{\hat{A}} \)
4. \( [\lambda x.M]_{\hat{A}} = f \) where the function \( f \) is defined by \( f(d) = [M]_{\hat{A}}^\iota \)
5. \( [\text{fst} M]_{\hat{A}} = \text{fst}([M]_{\hat{A}}) \)
6. \( [\text{snd} M]_{\hat{A}} = \text{snd}([M]_{\hat{A}}) \)
7. \( [M_1, M_2]_{\hat{A}} = ([M_1]_{\hat{A}}, [M_2]_{\hat{A}}) \)
8. \( [\text{R}_\iota(G, F, N)]_{\hat{A}} = f(n) \) where \( n = [N]_{\hat{A}} \) and the function \( f \) is defined by \( f(k+1) = [F]_{\hat{A}}^\iota (f(k)) \) and \( f(0) = [G]_{\hat{A}}^\iota \)
9. \( [Y_\sigma M]_{\hat{A}} = f^\iota(0_\sigma) \) where \( k = [\sigma]_b \), the function \( f \) is defined by \( f = [M]_{\hat{A}} \) and \( 0_\sigma \) is some fixed element in \( T^b_{\sigma} \). (We explain the role of this fixed element towards the end of this section.)
Lemma 5. For any type $\sigma$ and any $t \in T^\circ_\rho$, we have $\bot \preceq_\sigma t$.

Proof. We prove this lemma by induction on structure of $\sigma$. The statement follows trivially from the definition of $\preceq_\sigma$ when $\sigma = t$. Assume $\sigma = \rho \rightarrow \tau$. Fix an arbitrary $t$ in $T^\circ_\rho$. The definition of $\preceq_{\rho \rightarrow \tau}$ states that $\bot_{\rho \rightarrow \tau} \preceq_{\rho \rightarrow \tau} t$ holds if $\bot_{\rho \rightarrow \tau}(i) \preceq_{\tau} t(j)$ for any $i \preceq_\rho j$. By the definition of $\bot_{\rho \rightarrow \tau}$, we have $\bot_{\rho \rightarrow \tau}(i) = \bot_\tau$ for any $i \in D^\circ_\rho$. By the induction hypothesis on $\tau$, we have $\bot_\tau \preceq_{\tau} t(j)$ for any $j \in T^\circ_\rho$. Hence, we have $\bot_{\rho \rightarrow \tau}(i) = \bot_\tau \preceq_{\tau} t(j)$ for any $i \in D^\circ_\rho$ and any $j \in T^\circ_\rho$, and particularly, we have $\bot_{\rho \rightarrow \tau}(i) \preceq_{\tau} t(j)$ for any $i, j$ such that $i \preceq_\rho j$. Thus, the relation $\bot_{\rho \rightarrow \tau} \preceq_{\rho \rightarrow \tau} t$ holds. We omit the case when $\sigma = \rho \otimes \tau$.

Lemma 6. Let $d \in D^\circ_\rho$, $d' \in D^\circ_\rho$, $t \in T^\circ_\rho$, and $t' \in T^\circ_\rho$. If $d \preceq_{\rho \rightarrow \tau} t$ and $d' \preceq_\rho t'$, then $d(d') \preceq_{\tau} t(t')$.

Proof. This follows straightforwardly from the definition of $\preceq_{\rho \rightarrow \tau}$. The definition states that $d \preceq_{\rho \rightarrow \tau} t$ iff we have $d(i) \preceq_{\tau} t(j)$ for any $i \preceq_\rho j$. Hence, we have $d(d') \preceq_{\tau} t(t')$ if $d \preceq_{\rho \rightarrow \tau} t$ and $d' \preceq_\rho t'$.

Lemma 7. For any assignment $\mathcal{A}$, any $\mathcal{A}$-compatible total assignment $\hat{\mathcal{A}}$, and any $\text{PCF}_\rho^\circ$-term $M$ of type $\sigma$, we have $\sem{M}_\mathcal{A} \preceq_\sigma \sem{\hat{M}}_{\hat{\mathcal{A}}}$.

Proof. We prove this lemma by induction on the structure of $M$. Case $M \equiv k_n$: We have $\sem{k_n}_{\mathcal{A}} \preceq_i \sem{k_n}_{\hat{\mathcal{A}}}$ by the definition of $\preceq_i$. Case $M \equiv x^a$: We have $\sem{x}_{\mathcal{A}} = \mathcal{A}(x) \preceq_\sigma \hat{\mathcal{A}}(x) = \sem{x}_{\hat{\mathcal{A}}}$ since the assignment $\hat{\mathcal{A}}$ is $\mathcal{A}$-compatible.

Case $M \equiv (M_1 \ldots M_k)$: We have $\sem{M_1}_{\mathcal{A}} \preceq_{\rho \rightarrow \tau} \sem{M_1}_{\hat{\mathcal{A}}}$ by induction hypothesis on $M_1$, and we have $\sem{M_2}_{\mathcal{A}} \preceq_{\rho \rightarrow \tau} \sem{M_2}_{\hat{\mathcal{A}}}$ by induction hypothesis on $M_2$. Hence, we have $\sem{(M_1 M_2)}_{\mathcal{A}} = \sem{M_1}_{\mathcal{A}}(\sem{M_2}_{\mathcal{A}}) \preceq_{\tau} \sem{M_1}_{\mathcal{A}}(\sem{M_2}_{\hat{\mathcal{A}}}) = \sem{(M_1 M_2)}_{\hat{\mathcal{A}}}$ by Lemma 6.

Case $M \equiv R_\sigma(G, F, N)$: The proof splits into the two cases (i) $\sem{N}_{\mathcal{A}} = \bot$ and (ii) $\sem{N}_{\mathcal{A}} = n \neq \bot$. In Case (i) we have $\sem{R_\sigma(G, F, N)}_{\mathcal{A}} = \bot$, and thus, $\sem{R_\sigma(G, F, N)}_{\mathcal{A}} \preceq_\sigma \sem{R_\sigma(G, F, N)}_{\hat{\mathcal{A}}}$ by Lemma 5. We turn to Case (ii). By induction hypothesis on $N$, we have $n = \sem{N}_{\mathcal{A}} \preceq_\sigma \sem{N}_{\hat{\mathcal{A}}}$. This entails that $\sem{N}_{\mathcal{A}} = n$. By the definitions of $\sem{\cdot}_{\mathcal{A}}$ and $\sem{\cdot}_{\hat{\mathcal{A}}}$, we have $\sem{R_\sigma(G, F, N)}_{\mathcal{A}} = \sem{F}_{\mathcal{A}}(n - 1, \sem{F}_{\mathcal{A}}(n - 2, \ldots \sem{F}_{\mathcal{A}}(0, \sem{G}_{\mathcal{A}})))$ and $\sem{R_\sigma(G, F, N)}_{\hat{\mathcal{A}}} = \sem{F}_{\hat{\mathcal{A}}}(n - 1, \sem{F}_{\hat{\mathcal{A}}}(n - 2, \ldots \sem{F}_{\hat{\mathcal{A}}}(0, \sem{G}_{\hat{\mathcal{A}}}))$.

Use a sub induction on $n$ to prove $\sem{R_\sigma(G, F, N)}_{\mathcal{A}} \preceq_\sigma \sem{R_\sigma(G, F, N)}_{\hat{\mathcal{A}}}$. 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** \( M \equiv Y_\sigma F \): It suffices to prove that we have

\[
\llbracket F \rrbracket^k_A(\perp) \preceq_\sigma \llbracket F \rrbracket^k_A(0_\sigma) \quad (\dagger)
\]

for any \( k \in \mathbb{N} \). We prove \((\dagger)\) by sub induction 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** \( M \equiv \lambda x^\tau N^\tau \): We will prove that \([\lambda xN]_A(d) \preceq_\tau [\lambda xN]_A(t)\) whenever \( d \preceq_\rho t \). (Then we have \([\lambda xN]_A \preceq_\rho \tau [\lambda xN]_A\) by the definition of \( \preceq_\rho \).) Fix \( d \in D^\rho_\delta \) and \( t \in T^\rho_\delta \) such that \( d \preceq_\rho t \). The assignment \( A_\delta^\rho \) is \( A^\rho_\delta \)-compatible since the assignment \( A \) is \( A \)-compatible. By the induction hypothesis on \( N \), we have

\[
[\lambda xN]_A(d) = [N]_{A_\delta^\rho} \preceq_\tau [N]_{A_\delta^\rho} = [\lambda xN]_A(t).
\]

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

**Theorem 6 (Soundness II).** For any closed PCF\(_b\)\(^-\) term \( M \) of type \( \iota \), we have \( \llbracket M \rrbracket = n \) whenever \( M \gg k_n \).

**Proof.** Assume \( M \gg k_n \). By Theorem 5 and Lemma 7, we have \( n = [M] \preceq_\iota [M] \). The definition of the relation \( \preceq_\iota \) states that

\[
n \preceq_\iota [M] \iff n = \perp \lor n = [M].
\]

Thus, since \( n \neq \perp \), we have \( [M] = n \). □

The interpretation \([\cdot] \) is adequate in the sense that we have \( M \gg k_n \) whenever \( [M] = n \). (We will not prove this fact as it is not needed later in the paper.) The interpretation \([\cdot, \cdot] \) is not adequate. For instance, \( [Y_\iota (\lambda x^\iota x)] = n \) for some \( n \in \mathbb{N} \), but it is not the case that \( Y_\iota (\lambda x^\iota x) \gg k_n \). When \( M \) does not normalise, the value of \( [M] \) will depend on the choice of \( 0_\sigma \) in the interpretation of the fixed-point terms. The definition states that \( [Y_\sigma N]_A = [N]_{A_\delta}^k(0_\sigma) \) where \( k \) is a specific natural number and \( 0_\sigma \) is some element in \( T^\rho_\delta \). Any choice of \( 0_\sigma \in T^\rho_\delta \) will do in the sense that the validness of Theorem 6 does not depend on the choice.

The interpretation \([\cdot, \cdot] \) is preserved by the rewrite rules, i.e., we have \([M]_A = [M']_A \) whenever \( M \gg M' \). In contrast, we might have \( [M]_A \neq [M']_A \) when \( M \gg M' \). 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, that is, Theorem 6 cannot be proved without invoking Theorem 5.
2.3 Interpretations in Finite Sets of Natural numbers

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

**Definition.** If $\alpha < |\sigma|_b$, then $\alpha$ can be viewed as a $|\sigma|_b$-digit number in base $|\tau|_b$, and hence, uniquely written in the form

$$v_0 + v_1|\tau|_b^1 + \ldots + v_k|\tau|_b^k$$

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 $\alpha$, and for any $i < |\sigma|_b$, we denote the $i$’th digit of $\alpha$ by $a[i]_b$, that is, $a[i]_b = v_i$. A *valuation* $V$ is a total function from the set of variables into $\mathbb{N}$ such that $V(x) < |\sigma|_b$ for any variable $x$ of type $\sigma$. We define the valuation $V^b_\tau$ by $V^b_\tau(x) = i$ and $V^b_\tau(y) = V(y)$ for any variable $y$ different from $x$. We define the *value of the term* $M$ *at the base* $b$ *under valuation* $V$, written $\text{val}^b_{\sigma}(M)$, recursively over the structure of $M$. Let

1. $\text{val}^b_{\sigma}(x) = V(x)$
2. $\text{val}^b_{\sigma}(\lambda x^\sigma M^\tau) = \sum_{i < |\sigma|_b} \text{val}^b_{\sigma}(M) \times |\tau|_b^i$
3. $\text{val}^b_{\sigma}(M \cdot N) = \text{val}^b_{\sigma}(M) + \text{val}^b_{\sigma}(N)$
4. $\text{val}^b_{\sigma}(\text{snd}(M^\tau)) = \text{val}^b_{\sigma}(M) \div |\tau|_b$ (integer division)
5. $\text{val}^b_{\sigma}(\text{fst}(M^\tau)) = \text{val}^b_{\sigma}(M) \mod |\tau|_b$
6. $\text{val}^b_{\sigma}(\text{snd}(M^\tau)) = \text{val}^b_{\sigma}(M)$ (mod $|\tau|_b$)
7. $\text{val}^b_{\sigma}(k_n) = n \; (\text{mod } |\sigma|_b)$
8. $\text{val}^b_{\sigma}(\text{Ro}(G,F,N)) = f(\text{val}^b_{\sigma}(N))$ where $f(k + 1) = (\text{val}^b_{\sigma}(F)[k]_b)[f(k)]_b$ and $f(0) = \text{val}^b_{\sigma}(G)$
9. $\text{val}^b_{\sigma}(\text{Y}(M)) = f^{\sigma|_b}(0)$ where $f(x) = \text{val}^b_{\sigma}(M)[x]_b$

For any closed term $M$, we have $\text{val}^b_{\sigma}(M) = \text{val}^b_{\sigma}(M)$ for any valuations $V_0, V_1$, and we will simply write $\text{val}^b_{\sigma}(M)$ when $M$ is closed.  

**Theorem 7 (Soundness III).** For any closed $\text{PCF}^b_{\sigma}$-term $\iota$ of type $\iota$, we have $\text{val}_{\sigma}(M) = n$ whenever $M \overset{\iota}{\Rightarrow} k_n$.

**Proof.** For any type $\sigma$, let $\mathbb{N}^\iota_0 = \{ n \mid n < |\sigma|_b \}$. The bijection $\iota_\sigma : T^b_{\sigma} \rightarrow \mathbb{N}^\iota_0$ is defined by

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

We will occasionally suppress the subscript $\sigma$ of the bijection $\iota_\sigma$, and we have

1. $\iota(t(t')) = \iota(t)[\iota(t')]_b$ for any $t \in T^b_{\rho \rightarrow \tau}$ and any $t' \in T^b_{\rho}$
2. $\iota(\text{fst}(t)) = \iota(t) \div |\tau|_b$ for any $t \in T^b_{\rho \circ \tau}$
3. $\iota(\text{snd}(t)) = \iota(t) \mod |\tau|_b$ for any $t \in T^b_{\rho \circ \tau}$. 


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

(Claim) Let \( A \) be any assignment such that \( A(x) \in T^b_\sigma \) for all \( x \) of type \( \sigma \). Furthermore, let \( V \) be the valuation given by \( V(x^\sigma) = \iota_\sigma(A(x)) \). Then we have \( \text{val}^b_\sigma(M) = \iota_\sigma([M]_A) \) for any term \( M \) of type \( \sigma \).

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 \( M \equiv \lambda x^\sigma N^\tau \) we have

\[
\text{val}^b_\sigma(\lambda x^\sigma N^\tau) = \sum_{i < |\sigma|_b} \text{val}^{\nu i}_b(N) \times |\tau|_b^i \\
= \sum_{i < |\sigma|_b} \iota([N]_{A_{i-1}(i)}) \times |\tau|_b^i \\
= \sum_{i < |\sigma|_b} \iota([\lambda x^\sigma N]_A) \times |\tau|_b^i \\
= \iota([\lambda x^\sigma N]_A) 
\]

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

Assume \( M \triangleright k_n \). By Theorem 6, we have \( [M]_n = n \). By the claim we have and the definition of \( \iota_i \), we have \( \text{val}_\sigma(M) = \iota_i([M]_n) = n \). \( \square \)

3 Proofs of the Main Results

The interpretation function \( \text{val} (\cdot) \) will serve two purposes. One purpose is rather obvious. A closed term \( M^\rightarrow \) is a total program if \( Mk_n \) normalises 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 normalising 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 \( \text{val}_b(Mk_n) \) where

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

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

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

\[
0_\sigma : t, \text{ Succ}_\sigma : t, \sigma \rightarrow \sigma , \text{ Pred}_\sigma : t, \sigma \rightarrow \sigma , \text{ Le}_\sigma : t, \sigma, \sigma \rightarrow t \text{ and Eq}_\sigma : t, \sigma, \sigma \rightarrow t
\]

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

(i) \( \text{val}_{b+1}(0_\sigma) = 0 \)
(ii) \( \text{val}_{b+1}(\text{Succ}_\sigma(k_b, M)) = \text{val}_{b+1}(M) + 1 \; \text{mod } |\sigma|_{b+1} \)
(iii) $\text{val}_{b+1}(\text{Pred}_\sigma(k_b, M)) = \text{val}_{b+1}(M) - 1 \pmod{|\sigma|_{b+1}}$

(iv) $L_{\sigma}(k_b, M_1, M_2) \triangleright \begin{cases} k_0 & \text{if } \text{val}_{b+1}(M_1) \leq \text{val}_{b+1}(M_2) \\ k_1 & \text{otherwise} \end{cases}$

(v) $E_{\sigma}(k_b, M_1, M_2) \triangleright \begin{cases} k_0 & \text{if } \text{val}_{b+1}(M_1) = \text{val}_{b+1}(M_2) \\ k_1 & \text{otherwise}. \end{cases}$

These arithmetical terms become helpful in Section 3.2 where we prove the inclusion $2_{n+1}^{\text{lin}} \subseteq P_{n+1}$ and the inclusions $2_{n}^{\text{space}} \subseteq G_{2n}$ and $2_{n+1}^{\text{time}} \subseteq \mathcal{G}_{2n+1}$.

3.1 Arithmetic in $T^-$ and $\text{PCF}^-$

Variants of the next few lemmas can also be found in [26], but we will repeat the core of the proofs here.

**Lemma 8 (Conditionals).** For any type $\sigma$ there exists a $T^-$-term

$$\text{Cond}_\sigma : \iota, \sigma, \sigma \rightarrow \sigma$$

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

**Proof.** We prove the lemma by induction on the structure of $\sigma$.

Let $\text{Cond}_\sigma \equiv \lambda x^\iota X. R_{\iota, \sigma}(y, \lambda u^\iota z, x)$. It is easy to see that the term $\text{Cond}_\sigma$ possesses the required properties.

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

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

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

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

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

Use the $\eta$-equality $(\text{fst} X, \text{snd} X) = X$ to prove that the term $\text{Cond}_\sigma$ possesses the required properties. We omit the details.

Only recursor terms of type $\iota$ occur in $\text{Cond}_\sigma$, and thus we have $\text{Rk}(\text{Cond}_\sigma) = 0$. □

**Lemma 9 (Long Iterations in $T^-$).** For all types $\sigma$ and $\tau$ there exists a $T^-$-term $\text{It}_\sigma^{\tau} : (\iota, \tau \rightarrow \tau, \tau) \rightarrow \tau$ such that $\text{It}_\sigma^{\tau}(k_0, F, G) = F^{\text{val}_{|\sigma|_{b+1}}}(G)$. Furthermore, we have $\text{Rk}(\text{It}_\sigma^{\tau}) = \text{dg}(\sigma) + \text{dg}(\tau)$. (We will call $\text{It}_\sigma^{\tau}$ an iterator.)
Proof. We prove the lemma by induction on the structure of \( \sigma \).

Assume \( \sigma = \iota \). Let \( I^\iota_r \equiv \lambda x:\text{Type} \rightarrow x. R(x, \lambda x. Y \sigma (x), \lambda x. Y \sigma (x)) \). Obviously, we have \( \text{Rk}(I_r^\sigma_\iota) = \text{dg}(\iota) + \text{dg}(\tau) \), and it is straightforward to prove by induction on \( b \) that \( I^\sigma_r(k_b, F; G) = F^b + 1(G) \). Thus, the lemma holds when \( \sigma = \iota \).

Assume \( \sigma = \sigma_1 \to \sigma_2 \). Let \( I^\sigma_r \equiv \lambda x:\text{Type} \rightarrow x. (I^\sigma_r_1(x, I^\sigma_2(x), Y \sigma (x), Y \sigma (x)) \). We have

\[
\text{Rk}(I^\sigma_r) = \max(\text{Rk}(I^\sigma_r_1), \text{Rk}(I^\sigma_r_2))
\]

\[
= \max(\text{dg}(\sigma_1) + \text{dg}(\tau), \text{dg}(\sigma_2) + \text{dg}(\tau))
\]

\[
= \max(\text{dg}(\sigma_1) + \text{dg}(\tau) + 1, \text{dg}(\sigma_2) + \text{dg}(\tau))
\]

\[
= \max(\text{dg}(\sigma_1) + 1, \text{dg}(\sigma_2) + \text{dg}(\tau))
\]

\[
= \text{dg}(\sigma) + \text{dg}(\tau).
\]

So, the iterator has the right rank. We will now prove that we indeed have \( I^\sigma_r(k_b, F; G) = F^{|\sigma|_{b+1}} G \). Let \( A \equiv (I^\sigma_r k_b) \). We prove by induction on \( k \) that \( (A^k F) G = F^{|\sigma|_{b+1}} G \). We have \( (A^0 F) = F \), and hence \( (A^0 F) G = F^{|\sigma|_{b+1}} G \).

Moreover,

\[
(A^{k+1} F) G = (A (A^k F)) G = I^\sigma_r(k_b, A^k F; G) = (A^k F)^{|\sigma|_{b+1}} G = F^{|\sigma|_{b+1}} G.
\]

The two last equalities hold respectively by induction hypothesis on \( \sigma_2 \) and by induction hypothesis on \( k \). This proves \((*)\). Furthermore, we have

\[
I^\sigma_r(k_b, F; G) = I^\sigma_1(k_b, (I^\sigma_2 k_b), F) G
\]

\[
= (I^\sigma_2 k_b)^{|\sigma|_{b+1}} F G
\]

\[
= F^{|\sigma|_{b+1}} G
\]

\[
= F^{|\sigma|_{b+1}} G.
\]

When \( \sigma = \sigma_1 \times \sigma_2 \), let \( I^\sigma_r \equiv \lambda x:\text{Type} \rightarrow x. (I^\sigma_r_1(x, I^\sigma_2(x), Y \sigma (x), Y \sigma (x)) \). and the lemma holds. We omit the details. \( \square \)

Lemma 10 (Basic functions). The following number-theoretic functions can be defined by \( \text{T}^- \)-terms of rank 0. (i) \( x - y \) (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 \( f(x, m) = x + 1 \) (mod \( m + 1 \)) for \( x \leq m \).

Proof. The constant function 0 is defined by the initial \( \text{T}^- \)-term \( k_0 \). The projection function \( u^n_i(x_1, \ldots, x_n) = x_i \) is defined by the \( \text{T}^- \)-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 \( \text{T}^- \)-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 \( P(y + 1) = u^2_i(y, P(y)) \), and then (i) holds since \( x - 0 = 0 \) and \( x - (y + 1) = P(x - y) \). 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 \(c(x + 1, y_1, y_2) = u^2_2(y_1, y_2, c(x, y_1, y_2))\).

Finally, (iii) holds since \(c(m-x, 0, m-(m-x)-1)) = x + 1 \mod m\) for \(x \leq m\).

\[\square\]

**Lemma 11 (Arithmetic in \(T^-\)).** For any type \(\sigma\) there exists \(T^-\)-terms

\[0_\sigma : i, \text{Suc}_\sigma : i, \sigma \rightarrow \sigma, \text{Pred}_\sigma : i, \sigma \rightarrow \sigma, \text{Le}_\sigma : i, \sigma, \sigma \rightarrow i\]

of rank \(2dg(\sigma) - 2\) such that (i), (ii), (iii), (iv) and (v) at page 15 hold.

**Proof.** Let \(0_\sigma \equiv k_0; 0_\pi \equiv \langle 0_\sigma, 0_\tau \rangle\); and \(0_\pi \equiv \lambda x^\tau 0_\tau\). Obviously, we have \(\text{val}_{b+1}(0_\sigma) = 0\) and \(\text{Rk}(0_\sigma) = 0 \leq 2dg(\sigma) - 2\) for any \(\sigma\). Thus, (i) holds.

We will define \(\text{Suc}_\sigma\), \(\text{Le}_\sigma\) and \(\text{Eq}_\sigma\) in parallel recursively over the structure of \(\sigma\). We omit the definition of \(\text{Pred}_\sigma\) as this definition is very similar to the definition of \(\text{Suc}_\sigma\).

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 \(\sigma = \pi \rightarrow \tau\). We define \(F\) by

\[
F(x, y, z) = \lambda b^X Y^\sigma z^1 X^\pi \text{Cond}_1 \left(\text{Eq}_\pi(b, \text{snd}z, Y(\text{snd}z))\right) \\
\quad \text{fst} \left(\text{Suc}_\pi(b, \text{snd}z)\right), \text{Cond}_1 \left(\text{Le}_\pi(b, X(\text{snd}z), Y(\text{snd}z))\right) \\
\quad \text{fst}(k_0, \text{Suc}_\pi(b, \text{snd}z)), \text{fst}(k_1, \text{Suc}_\pi(b, \text{snd}z)) \right) .
\]

By the induction hypothesis, we have

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

Thus, we have

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

and then (iv) holds when

\[
\text{Le}_\sigma \equiv \lambda b^X Y^\sigma \text{fst} \text{Rk}_1^\tau(b, F(h, X, Y), \langle k_0, 0_\pi \rangle) .
\]

Now, let \(\text{Eq}_\sigma \equiv \lambda b^X Y^\sigma \text{Cond}_1(\text{Le}_\sigma(b, X, Y), \text{Cond}_1(\text{Le}_\sigma(b, Y, X), k_0, k_1), k_1)\)

and (v) holds. We will now argue that \(\text{Le}_\sigma\) and \(\text{Eq}_\sigma\) have the required rank. First, we note that

\[
\text{Rk}(F) \leq \max(2dg(\pi) - 2, 2dg(\tau) - 2)
\]
follows from Lemma 8 and the induction hypothesis. Furthermore, we have

\[
\text{Rk}(\text{Eq}_\sigma) = \text{Rk}(\text{Le}_\sigma) \\
= \max(\text{Rk}(F), \text{Rk}(\text{It}_{i \times \pi})) \\
= \max(\text{Rk}(F^i), \text{dg}(\pi) + \text{dg}(\iota \times \pi)) \\
\leq \max(\text{Rk}(F^i), \text{dg}(\pi) + \text{dg}(\iota \times \pi)) \\
= \max(2\text{dg}(\pi), 2\text{dg}(\tau) - 2) \\
= \max(2\text{dg}(\pi), 2\text{dg}(\tau) - 2) \\
= \max(2\text{dg}(\pi) + 2, 2\text{dg}(\tau) - 2) \\
= 2\max(\text{dg}(\pi) + 1, \text{dg}(\tau) - 2) \\
= 2\text{dg}(\sigma) - 2 .
\]

Thus, Eq\_\sigma and Le\_\sigma have the required rank. Next we define Suc\_\sigma. 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

\[
a + 1 = v_0|\tau|_b^0 + \cdots + v'_i|\tau|_b^i + v_{i+1}|\tau|_b^{i+1} + \cdots + v_k|\tau|_b^k \quad \text{(mod } |\sigma|_b) \]

where \( v'_j = v_j + 1 \) (mod \( |\tau|_b \)) for \( j = 0, \ldots, i \). We call such an \( i \) the carry border of the number \( a \). Let

\[
C_\sigma \equiv \lambda b. \text{snd} \ \text{lt}_{i \times \pi}(b, G(b, X), (k_0, 0_{\pi})) \quad (\dagger)
\]

where

\[
G \equiv \lambda b'. X^{\pi - \tau} z^{i \times \pi}. \text{Cond}(\text{fst} z, \text{Cond}_{i \times \pi}(\text{Eq}_\tau(b, \text{Suc}_\tau(X(\text{snd} z)), 0_{\pi}), (k_0, \text{Suc}_\tau(\text{snd} z)), (k_1, \text{Suc}_\tau(\text{snd} z)))).
\]

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

\[
\text{Suc}_\sigma \equiv \lambda b'. X^{\sigma \tau} z^{i \times \pi}. \text{Cond}(\text{le}_\tau(b, i, C_\sigma(b, X)), \text{Suc}_\tau(X(i)), X(i))
\]

and \( (i) \) holds. An argument similar to the one showing that the ranks of Eq\_\sigma and Le\_\sigma are bounded by \( 2\text{dg}(\sigma) - 2 \), will show that also the rank of C\_\sigma is bounded by \( 2\text{dg}(\sigma) - 2 \). The rank of Suc\_\sigma equals the rank of C\_\sigma.

Let \( \sigma = \pi \times \tau \). Define Suc\_\sigma such that

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

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

\[
(\text{Le}_\sigma(b, F, F') = k_0) \land \text{Eq}_\pi(b, F, F') \neq k_0) \lor \quad \text{(Le}_\sigma(b, G, G') = k_0) \land \text{Eq}_\pi(b, F, F') = k_0)
\]

Define Eq\_\sigma as above. It is easy to construct the required terms, and we skip the details. \( \square \)
Lemma 12 (Arithmetic in PCF\(^-\)). For any type \(\sigma\) there exists PCF\(^-\)-terms

\[
0_{\sigma}:\tau,\ Suc_{\sigma}:\tau,\ \sigma \rightarrow \tau,\ Pred_{\sigma}:\tau,\ \sigma \rightarrow \tau,\ Le_{\sigma}:\tau,\ \sigma,\ \sigma \rightarrow \tau
\]

of rank \(\text{dg}(\sigma)\) such that (i), (ii), (iii), (iv) and (v) at page 15 hold.

\textbf{Proof.} This proof is nearly identical to the proof of Lemma 11. We define the terms \(\text{Suc}_{\sigma}\), \(\text{Pred}_{\sigma}\), \(\text{Le}_{\sigma}\) and \(\text{Eq}_{\sigma}\) as we do in the proof of Lemma 11, i.e., in parallel recursively over the structure of \(\sigma\), 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 \(\sigma = \pi \rightarrow \tau\), the statement marked (i) in the proof of Lemma 11 defines the \(T^-\)-term \(\text{Le}_{\sigma}\) by \(\text{Le}_{\sigma} \equiv \lambda bXY.\text{fst} \text{It}_{\pi\times\pi}(b,F(b,X,Y),\langle k_0,0_{\pi}\rangle)\) where \(F\) is a term such that

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

and

\[
\text{Rk}(F) = \max(\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Suc}_{\pi}))
\]

Given a term \(F\) with these properties, we can define a PCF\(^-\)-term \(\text{Le}_{\sigma}\) by applying the fixed-point term \((Y_{\pi \rightarrow \iota \otimes \pi}A)\) where

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

Then we have

\[
(Y_{\pi \rightarrow \iota \otimes \pi}A)_{0_{\pi}} = F(k_b,X,Y)|_{\pi^{k+1}}(\langle k_0,0_{\pi}\rangle).
\]

Now, let \(\text{Le}_{\sigma}\) be the PCF\(^-\)-term given by \(\text{Le}_{\sigma} \equiv \lambda bXY.\text{fst}((Y_{\pi \rightarrow \iota \otimes \pi}A)_{0_{\pi}})\). It follows by (*) and induction hypothesis on \(\text{Eq}_{\pi}\) and \(\text{Suc}_{\pi}\) that clause (iv) of our lemma holds. By inspecting the construction of \(\text{Le}_{\sigma}\), we see that

\[
\text{Rk}(\text{Le}_{\sigma}) = \max(\text{Rk}(F),\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Suc}_{\pi}),\text{dg}(\pi \rightarrow \iota \otimes \pi)) \rightarrow
\max(\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Suc}_{\pi}),\text{dg}(\pi \rightarrow \iota \otimes \pi)) \rightarrow
\]

To verify that \(\text{Le}_{\sigma}\) has the required rank, we assume by induction hypothesis that

\[
\text{Rk}(\text{Eq}_{\pi}) = \text{dg}(\pi) \text{ and } \text{Rk}(\text{Eq}_{\pi}) = \text{Rk}(\text{Suc}_{\pi}) = \text{dg}(\pi).
\]

Then we have

\[
\text{Rk}(\text{Le}_{\sigma}) = \max(\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Suc}_{\pi}),\text{dg}(\pi \rightarrow \iota \otimes \pi)) \rightarrow
\max(\text{Rk}(\text{Eq}_{\pi}),\text{Rk}(\text{Suc}_{\pi}),\text{dg}(\pi) + 1) \rightarrow
\max(\text{dg}(\tau),\text{dg}(\pi) + 1) \rightarrow
\text{dg}(\sigma).
\]

Along this line, we can also find a PCF\(^-\)-term \(\text{Suc}_{\sigma}\) satisfying the lemma by eliminating the iterator \(\text{It}_{\pi \times \pi}\) from the formula marked (i) in the proof of Lemma 11. The definition of \(\text{Pred}_{\sigma}\) is similar to the definition of \(\text{Suc}_{\sigma}\), and \(\text{Eq}_{\sigma}\) is defined straightforwardly from \(\text{Le}_{\sigma}\). \(\square\)
3.2 Simulating Turing Machines in \( T^-\) and \( \text{PCF}^-\)

**Theorem 8.** Let \( \sigma \) and \( \pi \) be types, and let \( m \) be a Turing machine running in space \( s \) and time \( t \) where \( s(|x|) \leq \max(|x|, 2) \) and \( t(|x|) \leq \max(|x|, 2) \). (i) There exists a \( T^-\)-term \( T^m_{\pi, \sigma} \) of rank \( \max(\text{dg}(\pi), \text{dg}(\sigma)) + 1 \) such that \( T^m_{\pi, \sigma}(k_x) \triangleright k_0 \) if \( m \) accepts \( x \); and \( T^m_{\pi, \sigma}(k_x) \triangleright k_1 \) if \( m \) rejects \( x \). (ii) There exists a \( \text{PCF}^-\)-term \( \text{PCF}^m_{\pi, \sigma} \) of rank \( \max(\text{dg}(\pi), \text{dg}(\sigma)) + 1 \) such that \( \text{PCF}^m_{\pi, \sigma}(k_x) \triangleright k_0 \) if \( m \) accepts \( x \); and \( \text{PCF}^m_{\pi, \sigma}(k_x) \triangleright k_1 \) if \( m \) rejects \( x \).

**Proof.** Let \( \{a_0, \ldots, a_n\} \) and \( \{q_0, \ldots, q_l\} \) be respectively \( m \)'s alphabet and \( m \)'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 \( m \) by a closed term \( \tau(S', (T^{\sigma^{-1}}T^\sigma)) \) of type \( \tau \cdot \xi \cdot (\sigma \rightarrow \iota \cdot \sigma) \) where \( S \) represents the current state, \( T \) represents the tape, and \( \iota \) is the position of the head.

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

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

\[
\text{Md} \equiv \lambda F^{\sigma^{-1}}X^\sigma Y^\sigma. \text{Cond}_1(\text{Eq}_\sigma(k_x, X, Y), V, F(Y))
\]

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

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

To execute sufficiently many of \( m \)'s transitions in \( T^- \), we apply the iterator \( \text{It}_\xi \), that is, the \( T^- \)-term given by Lemma 9. We have

\[
\text{It}_\xi^\sigma(k_x, \text{Step}_\sigma(k_x), \text{Init}_\sigma(k_x)) = \text{Step}_\sigma(k_x)^{\text{val}_x(\text{Init}_\sigma(k_x))}
\]

and thus, the term simulates since \( m \) runs in time \( |\pi|_x \). Let

\[
T^m_{\pi, \sigma} \equiv \lambda y. \text{fst} \text{It}_\xi^\sigma(y, \text{Step}_\sigma(y), \text{Init}_\sigma(y)).
\]

Then, \( T^m_{\pi, \sigma}(k_x) \triangleright k_0 \) if \( m \) accepts \( x \), and \( T^m_{\pi, \sigma}(k_x) \triangleright k_1 \) if \( m \) rejects \( x \). Furthermore, since \( \text{dg}(|\xi|) = \text{dg}(\sigma) + 1 \), we have

\[
\text{Rk}(T^m_{\pi, \sigma}) = \text{Rk}(\text{It}_\xi^\sigma) = \text{dg}(\pi) + \text{dg}(\xi) = \text{dg}(\pi) + \text{dg}(\sigma) + 1.
\]
This proves (i).

Next, assume we are working in PCF. Let $\text{PCF}_{\pi,\sigma}^m \equiv \lambda y^{\text{fst}}((Y_{\pi \rightarrow \xi} P)0_{\pi})$ where

$$P \equiv \lambda X\pi^{-\xi} Z\pi.\text{Cond}_\xi(Eq_n(y',\text{Suc}_\pi(y',z),0_\pi),\text{Init}_\sigma(y'),\text{Step}_\pi(y',XSuc(y',z))).$$

Then, $\text{PCF}_{\pi,\sigma}^m(k_x) \gg k_0$ if $m$ accepts $x$, and $\text{PCF}_{\pi,\sigma}^m(k_x) \gg k_1$ if $m$ rejects $x$. Furthermore,

$$\text{Rk}(\text{PCF}_{\pi,\sigma}^m) = \text{dg}(\pi \rightarrow \xi) = \max(\text{dg}(\pi) + 1, \text{dg}(\xi)) = \max(\text{dg}(\pi) + 1, \text{dg}(\sigma \rightarrow i)) = \max(\text{dg}(\pi), \text{dg}(\sigma)) + 1.$$

This proves (ii) $\square$

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

**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

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

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

$$2^{(2x+2)^k} = 2^{2^{|x|}_{n+1}} \leq 2^{2^{\log_2(2x+2)^k}} \leq (|x|^{|x|}_{\text{max}(x,2)})^{|x|_{\text{max}(x,2)}} = |\sigma|_{\ell^{|x|}_{\text{max}(x,2)}}$$

and $\text{dg}(\sigma \rightarrow i) = n + 1$. $\square$

**Theorem 9.** We have space $2^m_0 \subseteq G_2$ and time $2^m_{n+1} \subseteq G_{2n+1}$. Furthermore, we have time $2^{m}_{n+1} \subseteq P_{n+1}$.

**Proof.** Let $A \in \text{TIME } 2^m_{n+1}$, and let $m$ be a Turing machine which decides $A$ in time, and thus also in space, $2^{|x|}_{n+1}$ for some fixed number $k$. By Lemma 13, there exists a type $\sigma$ of degree $n$ such that $2^{|x|}_{n+1} \leq |\sigma|_{x}$. By Lemma 8, we have a $T^*$-term $T^*_{\pi,\sigma}$ that decides $A$ and a $\text{PCF}^*$-term $\text{PCF}_{\pi,\sigma}^m$ that decides $A$. According to the Lemma, the term $T^*_{\pi,\sigma}$ is of rank $\text{dg}(\sigma) + \text{dg}(\sigma) + 1$, that is, of rank $2n + 1$, whereas $\text{PCF}_{\pi,\sigma}^m$ is of rank $\max(\text{dg}(\sigma), \text{dg}(\sigma)) + 1$, that is, of rank $n + 1$. This proves time $2^m_{n+1} \subseteq G_{2n+1}$ and time $2^{m}_{n+1} \subseteq P_{n+1}$ for all $n \in \mathbb{N}$.

Let $A \in \text{SPACE } 2^m_{n+1}$. Thus, there exists a Turing machine $m$ deciding $A$ in space $2^{|x|}_{n+1}$ and time $2^{k^m_{|x|}}_{n+2}$ for some $k, k'$. By Lemma 13, there exist a type $\sigma$ of degree $n$ and a type $\rho$ of degree $n + 1$ such that $2^{|x|}_{n+1} \leq |\sigma|_{x}$ and $2^{k^m_{|x|}}_{n+2} \leq |\rho|_{x}$. By Lemma 8, we have a $T^*$-term $T^*_{\rho,\sigma}$ deciding $A$. The rank of $T^*_{\rho,\sigma}$ is $\text{dg}(\rho) + \text{dg}(\sigma) + 1 = 2n + 2$. The inclusion space $2^m_{0} \subseteq G_0$ requires a tailored proof, and we omit the details. $\square$
3.3 Evaluating PCF$^-$-Terms by Turing Machines

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

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

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

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

Lemma 15. For any type $\sigma$ of degree $n + 1$ there exists a polynomial $p$ such that $|\sigma|_x \leq 2^{p(x)}$.

Proof. First, we note that $|\iota|_x = 1$ and $|\rho \otimes \tau|_x = |\rho|_x + |\tau|_x$, and hence

$$|\sigma|_x$$

is a constant function when $\sigma$ is of degree 0. (*)

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

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

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

Let $V$ be any fixed valuation. The value $\text{val}_x^V(M)$ can be computed by a Turing machine running in time $2^{n+1}$ for some $k \in \mathbb{N}$. 
Proof. We will give an informal algorithm for computing the number $\text{val}_x^y(M)$ 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 bounded by $2^{p(x)}$ for some polynomial $p$. It is to easy to see that the informal algorithm can be implemented by a Turing machine $m$ running in time $2^{p_0(x)}$ for some polynomial $p_0$, and hence, there exists $k \in \mathbb{N}$ such that $m$ runs in time $2^{k|x|}$.

Let $y_1, \ldots, y_k$ 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 $v_i/a_1, \ldots, v_i/a_\ell$ where the natural number $a_i$ is the value currently assigned to the variable $y_i$. The number $\ell$ 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 $\sigma$. We will have $\text{dg}(\tau) \leq n + 1$ for any variable $y_i^\tau$ in the list, and by Lemma 14 there exists a polynomial $p_0$ such that $2^{p_0(x)}$ bounds every $a_i$ in the list. Thus, each $a_i$ can be represented by a bit string of length $2^{p_0(x)}$. 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 $2^{p(x)}$.

The algorithm computes the value $\text{val}_x^y(M)$ 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 $\text{val}_x^y(M) = m \pmod{x + 1}$, and the algorithm will simply output the number $m \pmod{x + 1}$. The number of steps required to complete this task is obviously bounded by $2^{p(x)}$ for some polynomial $p$.

Case $M \equiv y_i$. We have $\text{val}_x^y(M) = V(y_i)$, 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 $2^{p(x)}$ for some polynomial $p$.

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

\begin{verbatim}
sum:=0;
for i = 0, \ldots, |\pi|_x - 1 do { assign i to z; sum:= sum \times |\tau|_x + \text{val}_i^y(N); }
output sum
\end{verbatim}

We have $\text{dg}(\pi) \leq n$ since $\text{dg}(\sigma) = \max(\text{dg}(\pi) + 1, \text{dg}(\tau)) \leq n + 1$. By Lemma 14, there exist polynomials $p_0$ and $p_1$ such that $|\pi|_x \leq 2^{p_0(x)}$ and $|\sigma|_x \leq 2^{p_1(x)}$. Hence, the loop will be executed no more than $2^{p_0(x)}$ times. Furthermore, the number computed into the register $\text{sum}$ is less than $|\sigma|_x$ and thus bounded by $2^{p_1(x)}$. The number of bits required to represent the number is bounded by $2^{p_1(x)}$.

The induction hypothesis yields a polynomial $p_2$ such that the number of steps required to compute $\text{val}_x^y(N)$ is bounded by $2^{p_2(x)}$. It follows that there exists
a polynomial $p$ such that the number of steps required to compute $\text{val}^Y_s(M)$ is bounded by $2^p(x)$.

Case $M^\sigma \equiv Y_\sigma(N)$. We have $\text{val}^Y_s(Y_\sigma N) = f^{[\sigma]x}(0)$ where $f(y) = \text{val}^Y_s(N)[y]_x$.

The proof splits into the two cases: (i) $\text{dg}(\sigma) = n + 1$ and (ii) $\text{dg}(\sigma) \leq n$. We will give the proof for case (i). Case (ii) is easier, and we leave the proof to the reader.

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

\begin{verbatim}
a:=0; for i = 0, ..., [\sigma]_x do { assign a to z; a:=\text{val}^Y_s(P) }
\end{verbatim}

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

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

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

\begin{theorem}
We have $\mathcal{P}_{n+1} \subseteq \text{TIME } 2^{n+1}.$
\end{theorem}

\begin{proof}
Assume $A \in \mathcal{P}_{n+1}$. The definition of $\mathcal{P}_{n+1}$ says that there exists a closed PCF*-term $M^{\rho \rightarrow \sigma}$ of rank $n + 1$ such that $Mk_x \not\rightarrow k_0$ if $x \notin A$, and $Mk_x \not\rightarrow k_1$ if $x \in A$. By applying the reduction rules of the typed $\lambda$-calculus, i.e. $\alpha$-conversions and $\beta$-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 $f_A(x) = \text{val}^Y_{\text{max}(x,m)+1}(Nk_x)$ 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 $2^{k_{\text{max}(x,m)+1}}$ for some $k \in \mathbb{N}$, and thus, in time $2^{n+1}$ for some $k' \in \mathbb{N}$. Hence, we have $A \in \text{TIME } 2^{n+1}$ and the inclusion $\mathcal{P}_{n+1} \subseteq \text{TIME } 2^{n+1}$ holds.
\end{proof}

References


