Theories with self-application and computational complexity

Thomas Strahm[*]

Version of May 2002

Final version to appear in **Information and Computation**

Abstract

Applicative theories form the basis of Feferman's systems of explicit mathematics, which have been introduced in the early seventies. In an applicative universe, all individuals may be thought of as operations, which can freely be applied to each other: self-application is meaningful, but not necessarily total. It has turned out that theories with self-application provide a natural setting for studying notions of abstract computability, especially from a proof-theoretic perspective.

This article is concerned with the study of (unramified) *bounded* applicative theories which have a strong relationship to classes of computational complexity. We propose new applicative systems whose provably total functions coincide with the functions computable in polynomial time, polynomial space, polynomial time *and* linear space, as well as linear space. Our theories can be regarded as applicative analogues of traditional systems of bounded arithmetic. We are also interested in higher type features of our systems; in particular, it is shown that Cook and Urquhart's system PV^ω is directly contained in a natural applicative theory of polynomial strength.

1 Introduction

Theories with self-application form the operational core of Feferman's systems of *explicit mathematics*, which have been introduced in [24, 25, 26]. The

[*]Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland. Email: `strahm@iam.unibe.ch`

original aim of explicit mathematics was to provide a logical basis for Bishop-style constructive mathematics. More generally, the explicit framework has gained considerable importance in proof theory in connection with the proof-theoretic analysis of subsystems of second order arithmetic and set theory. In particular, it was possible to reduce prima-facie non-constructive systems to a constructively justifiable framework. The most famous example in this connection is the reduction of the subsystem of second order arithmetic based on Δ^1_2 comprehension and bar induction to the most prominent framework of explicit mathematics, T_0, achieved by Jäger [42] and Jäger and Pohlers [44]. The language and axioms of explicit mathematics have also been shown to provide a logical framework for functional and object-oriented programming (cf. e.g. [27, 28, 29, 66]).

In a typical formulation of explicit mathematics one has to deal with two sorts of objects, namely *operations* and *types*. It has turned out that already the operational or applicative core of explicit mathematics – so-called *applicative theories* – are of significant interest. In particular, applicative theories provide a natural framework for a proof-theoretic approach to abstract computations. In contrast to traditional formalizations of mathematics which follow a set-theoretic paradigm and an extensional approach to functions, applicative theories and explicit mathematics focus on an intensional point of view. In an applicative universe of discourse, all objects may be regarded as operations (or rules) which can be freely applied to each other; self-application is meaningful, though not necessarily total. The key example of such a domain are the (codes of) partial recursive functions, which form a partial combinatory algebra via the usual notion of partial recursive function application. Indeed, the axioms for an untyped partial combinatory algebra will be at the heart of all applicative theories discussed in this paper. As usual, they guarantee full recursion in an abstract and elegant way. The reader is referred to the article Jäger, Kahle and Strahm [43] for a recent survey and references on applicative theories.

The main purpose of the present contribution is the study of (unramified) *bounded* applicative systems which have a strong relationship to classes of *computational complexity*. We provide new applicative theories whose provably total functions coincide with the functions computable in *polynomial time*, *polynomial space*, simultaneously *polynomial time and linear space*, as well as *linear space*. Our theories can be seen as natural applicative analogues of systems of bounded arithmetic, cf. Buss [8, 9, 11], Hájek and Pudlák [37], and Krajíček [48].

The most famous theory of bounded arithmetic is Buss' S^1_2 (cf. [8]) or, equivalently, Ferreira's PTCA^+ (cf. [33]). Both systems are first order systems of arithmetic closely related to the polynomial time computable functions. Whereas the former theory is formulated over the natural numbers, the latter directly refers to the collection of finite words over $\{0,1\}$. Canonical extensions of S^1_2 are Buss' theories S^i_2 ($i \in \mathbb{N}$), which characterize the levels of the polynomial time hierarchy. Takeuti [67] has studied weak second order theories U^{i,w^*}_2 which are essentially equivalent to S^i_2. Moreover, Zambella [71] has set up a very appealing second order bounded arithmetic BA whose fragments are naturally related to the theories S^i_2. Further, in his seminal thesis [8], Buss studies second order bounded arithmetic theories U^1_2 and V^1_2 for the polynomial space and exponential time computable functions, respectively. For extensive lists of references on bounded arithmetic, see the monographs and survey articles cited above.

Apart from the area of bounded arithmetic, there is the rapidly growing field of *implicit computational complexity* and *tiered formal systems.* We refer the reader to the conclusion of this paper for some references and discussion concerning this area of research.

The framework of all our applicative theories is very uniform and simple. The various applicative systems studied in this article will only differ with respect to the available initial functions or functionals as well as principles of induction. Due to the high expressive power of the underlying applicative language, lower bound arguments will be considerably simpler than in traditional systems of bounded arithmetic. This leads, in particular, to a straightforward characterization of the class *simultaneous polytime and linspace*, something that has yet to be accomplished for traditional (non-self-applicative) bounded arithmetic. The upper proof-theoretic bounds for our systems, which are all based on *classical* logic, will be established by combining partial cut elimination with a suitable notion of realizability or witnessing.

For example, we will set up an applicative theory PT for the polynomial time computable functions. In PT we have available full recursion and, hence, terms for all partial recursive functions, e.g., exponentiation; however, convergence or totality can only be derived for terms defining polynomial time computable functions. We will see that Buss' S^1_2 or Ferreira's PTCA^+ are directly interpretable in PT. In addition, "bootstrapping" the polynomial time computable functions in PT is very pleasant and coding-free.

We will also be interested in higher type aspects of our applicative systems. It turns out that higher types arise very naturally in our applicative framework, and again the question arises which *functionals* provably converge in a given axiomatic setting. In this article it is shown that Cook and Urquhart's system PV^ω (cf. [23]), a natural higher type version of Cook's PV (cf. [21]), is *directly* contained in PT. The terms of PV^ω define exactly the so-called *Basic Feasible Functionals*, BFF. The BFF's have turned out to be a rather robust class of higher type functionals with many interesting characterizations, see Section 5 of this paper for further information and references.

Let us now give a quick guided tour through our paper. We start in Section 2 with a short review of known recursion-theoretic characterizations of various function complexity classes on the binary words $\mathbb{W}$ by means of bounded recursion on notation as well as bounded unary recursion. The so-obtained machine-independent characterizations will be crucial for lower as well as upper bound arguments used in the sequel of the paper.

In Section 3 we set up the central applicative framework. We start with introducing the basic theory B of operations and words and recall some of its crucial properties. Then we present various forms of bounded induction and define the four central systems of this article, PT, PS, PTLS, and LS, corresponding to the functions computable in polynomial time, polynomial space, polynomial time *and* linear space, as well as linear space, respectively.

In Section 4 we provide lower bound arguments for our applicative systems, i.e., we show that the functions from the respective function complexity classes are provably total in the four applicative theories mentioned above. In particular, we will see that forms of bounded recursion are very naturally derived by means of the fixed point theorem and exploiting our various principles of bounded induction.

Higher type issues are at the heart of Section 5. There we will recapitulate an intensional and an extensional version of the Cook-Urquhart system PV^ω and show that both systems are naturally contained in our applicative system PT for the polynomial time computable functions. Indeed, the embeddings presented in this section also give rise immediately to higher type systems corresponding to PS, PTLS, and LS.

Upper bounds for the four systems PT, PS, PTLS, and LS are established in Section 6. The upper bound arguments proceed in two steps. First, standard *partial cut elimination* is employed in a sequent version of our systems in order to show that derivations of sequents of positive formulas can be restricted to positive cuts. The second crucial step consists in establishing very uniform *realizability* theorems for our four systems, where a notion of realizability for positive formulas in the standard open term model $\mathcal{M}(\lambda\eta)$ is used. The spirit of our realizability theorems is related to the work of Leivant [49], Schlüter [58], and Cantini [16, 18] and, in fact, our notion of realizability can be seen as an applicative analogue of Buss' witnessing method, see [8, 10].

In Section 7 we present further natural applicative systems for various classes of computable functions. In particular, we will study a system PH which is closely related to the *polynomial time hierarchy*; the crucial axiom of PH is a very uniform type two functional π for bounded quantification. Further investigations in this section concern applicative theories whose provably total functions are exactly the primitive recursive functions.

The paper ends with concluding remarks concerning the systems and results of this paper as well as directions for future research.

A preliminary version of this paper has been circulated in May 2000. Moreover, the main bulk of this article is contained in Part II of the author's habilitation thesis [64]. Recently, Cantini [12] has studied substantial extensions of the theory PT by an axiom of choice for operations and a uniformity principle, both restricted to positive conditions. In addition, he has considered a form of self-referential truth, providing a fixed point theorem for predicates. Cantini shows in [12] that the recursive content of PT is not altered by adding all the mentioned principles. The methods of proof used by Cantini provide new general insight into the relationship between classical and intuitionistic applicative systems. Finally, in [18] Cantini has also studied an applicative theory based on safe induction in the spirit of implicit computational complexity, see also our remarks in the conclusion of this paper.

2 Recursion-theoretic characterizations of complexity classes

In this section we review know recursion-theoretic characterizations of various classes of computational complexity. We will work over the set of binary words $\mathbb{W} = \{0, 1\}^*$. Our main interest in the sequel are the functions on $\mathbb{W}$ which are computable on a Turing machine in *polynomial time*, *simultaneously polynomial time and linear space*, *polynomial space*, and *linear space*. In the following we let FPtime, FPtimeLinspace, FPspace, and FLinspace denote the respective classes of functions on binary words $\mathbb{W}$. For an extensive discussion of recursion-theoretic or function algebra characterizations of complexity classes the reader is referred to the survey article Clote [19].

We are interested in various kinds of successor operations on the binary words $\mathbb{W}$. As usual, s_0 and s_1 denote the binary successor functions which concatenate 0 and 1 to the end of a given binary word, respectively. We are also given a unary lexicographic successor s_ℓ on $\mathbb{W}$, which satisfies for all x

in $\mathbb{W}$ the following recursion equations,

$$\mathsf{s}_\ell(\epsilon) = 0, \qquad \mathsf{s}_\ell(\mathsf{s}_0 x) = \mathsf{s}_1 x, \qquad \mathsf{s}_\ell(\mathsf{s}_1 x) = \mathsf{s}_0(\mathsf{s}_\ell x).$$

We have used here ϵ to denote the empty word. Observe that s_ℓ is the successor operation in the natural wellordering $<_\ell$ of $\mathbb{W}$ according to which words are ordered by length and words of the same length are ordered lexicographically. Thinking of binary words as binary representations of natural numbers, s_ℓ essentially corresponds to the usual successor operation on the natural numbers. Finally, we let $*$ and $\times$ stand for the binary operations of *word concatenation* and *word multiplication*, respectively, where $x \times y$ denotes the word x, length of y times concatenated with itself.

Towards a function algebra characterization of the complexity classes mentioned above, we now want to introduce two schemas of *bounded recursion.* For that purpose, let G, H_0, H_1 and K be given functions on binary words of appropriate arities. We say the function F is defined by *bounded recursion on notation* (BRN) from G, H_0, H_1 and K, if

$$\begin{array}{rcll} F(\vec{x}, \epsilon) & = & G(\vec{x}), & \\ F(\vec{x}, \mathsf{s}_i y) & = & H_i(\vec{x}, y, F(x, \vec{y})), & \qquad (i = 0, 1) \\ F(\vec{x}, y) & \leq & K(\vec{x}, y) & \end{array}$$

for all $\vec{x}, y$ in $\mathbb{W}$. Here $x \leq y$ signifies that the length of the word x is less than or equal to the length of the word y. On the other hand, a function F is defined by *bounded lexicographic recursion* (BRL) from G, H and K, if

$$\begin{array}{rcl} F(\vec{x}, \epsilon) & = & G(\vec{x}), \\ F(\vec{x}, \mathsf{s}_\ell y) & = & H(\vec{x}, y, F(x, \vec{y})), \\ F(\vec{x}, y) & \leq & K(\vec{x}, y) \end{array}$$

for all $\vec{x}, y$ in $\mathbb{W}$. Hence, the crucial difference between bounded recursion on notation (BRN) and bounded lexicographic or unary recursion (BRL) is that the former recursion scheme acts along the *branches of the binary tree*, whereas the latter form of bounded recursion is with respect to the lexicographic ordering of the *full binary tree.*

In the following we use the notation of Clote [19] for a compact representation of function algebras. Accordingly, we call (partial) mappings from functions on $\mathbb{W}$ to functions on $\mathbb{W}$ *operators.* If $\mathcal{X}$ is a set of functions on $\mathbb{W}$ and OP is a collection of operators, then $[\mathcal{X}; \mathsf{OP}]$ is used to denote the smallest set of functions containing $\mathcal{X}$ and closed under the operators in OP. We call $[\mathcal{X}; \mathsf{OP}]$ a *function algebra.* Our crucial examples of operators in the sequel are (BRN)

and (BRL). A further operator is the *composition operator* (COMP), which takes functions $F, G_1, \dots, G_n$ and maps them to the usual composition of F with $G_1, \dots, G_n$. Below we also use I for the usual collection of projection functions and we simply write ϵ for the 0-ary function being constant to the empty word ϵ.

We are now ready to state the function algebra characterizations of the four complexity classes which are relevant in this paper. The characterization of FPtime is due to Cobham [20]. The delineations of FPtimeLinspace and FPspace are due to Thompson [68]. Finally, the fourth assertion of our theorem is due to Ritchie [56]. For a uniform presentation of all these results we urge the reader to consult Clote [19].

Theorem 1 *We have the following function algebra characterizations of the complexity classes mentioned above:*

1. $[\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *, \times; \mathsf{COMP}, \mathsf{BRN}] = \text{FPtime}$.
2. $[\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *; \mathsf{COMP}, \mathsf{BRN}] = \text{FPtimeLinspace}$.
3. $[\epsilon, \mathsf{I}, \mathsf{s}_\ell, *, \times; \mathsf{COMP}, \mathsf{BRL}] = \text{FPspace}$.
4. $[\epsilon, \mathsf{I}, \mathsf{s}_\ell, *; \mathsf{COMP}, \mathsf{BRL}] = \text{FLinspace}$.

Let us mention that indeed word concatenation $*$ is redundant in the presence of word multiplication $\times$, and we have included it in the formulation of this theorem for reasons of uniformity only.

3 The applicative framework

In this section we will introduce the applicative systems which will be relevant in the rest of this paper. We start with a precise description of the basic theory of operations and words B. Later we will discuss two basic forms of bounded induction which will be used to set up the central applicative frameworks PT, PTLS, PS, and LS.

3.1 The basic theory B of operations and words

All applicative systems to be considered below are formulated in the language $\mathcal{L}$; it is a language of partial terms with *individual variables* $a, b, c, x, y, z, u, v, f, g, h, \dots$ (possibly with subscripts). $\mathcal{L}$ includes *individual constants* k, s (combinators), $\mathsf{p}, \mathsf{p}_0, \mathsf{p}_1$ (pairing and unpairing), d_W (definition by cases on binary words), ϵ (empty word) $\mathsf{s}_0, \mathsf{s}_1$ (binary successors), p_W (binary predecessor), $\mathsf{s}_\ell, \mathsf{p}_\ell$ (lexicographic successor and predecessor), $\mathsf{c}_\subseteq$ (initial subword relation) and l_W (tally length of binary words). We also assume that the two constants $*$ (word concatenation) and $\times$ (word multiplication) belong to $\mathcal{L}$, however, not all our applicative systems will have axioms about $*$ and $\times$. Finally, $\mathcal{L}$ has a binary function symbol $\cdot$ for (partial) term application, unary relation symbols $\downarrow$ (defined) and W (binary words) as well as a binary relation symbol $=$ (equality).

The *terms* $r, s, t, \dots$ of $\mathcal{L}$ (possibly with subscripts) are inductively generated from the variables and constants by means of application $\cdot$. We write ts instead of $\cdot(t, s)$ and follow the standard convention of association to the left when omitting brackets in applicative terms. As usual, (s, t) is a shorthand for $\mathsf{p}st$. Moreover, we use the abbreviations 0 and 1 for $\mathsf{s}_0\epsilon$ and $\mathsf{s}_1\epsilon$, respectively. Furthermore, we write $s \subseteq t$ instead of $\mathsf{c}_\subseteq st = 0$ and $s \leq t$ for $\mathsf{l}_\mathsf{W}s \subseteq \mathsf{l}_\mathsf{W}t$; $s \subset t$ and $s < t$ are understood accordingly. Finally, $s{*}t$ stands for $*st$, and $s{\times}t$ for $\times st$.

The *formulas* $A, B, C, \dots$ of $\mathcal{L}$ (possibly with subscripts) are built from the atomic formulas $(s = t)$, $s\downarrow$ and $\mathsf{W}(s)$ by closing under negation, disjunction, conjunction, implication, as well as existential and universal quantification over individuals.

Our conventions concerning substitutions are as follows. As usual we write $t[\vec{s}/\vec{x}]$ and $A[\vec{s}/\vec{x}]$ for the substitution of the terms $\vec{s}$ for the variables $\vec{x}$ in the term t and the formula A, respectively. In this connection we often write $A(\vec{x})$ instead of A and $A(\vec{s})$ instead of $A[\vec{s}/\vec{x}]$.

Our applicative theories are based on *partial* term application. Hence, it is not guaranteed that terms have a value, and $t\downarrow$ is read as t *is defined* or t *has a value.* The *partial equality relation* $\simeq$ is introduced by

$$s \simeq t \ := \ (s\downarrow \vee t\downarrow) \to (s = t).$$

In the following we will use the following natural abbreviations concerning the predicate W ($\vec{s} = s_1, \dots, s_n$):

$$\begin{array}{rcl}
\vec{s} \in \mathsf{W} & := & \mathsf{W}(s_1) \wedge \cdots \wedge \mathsf{W}(s_n), \\
(\exists x \in \mathsf{W})A & := & (\exists x)(x \in \mathsf{W} \wedge A), \\
(\forall x \in \mathsf{W})A & := & (\forall x)(x \in \mathsf{W} \to A), \\
(\exists x \leq t)A & := & (\exists x \in \mathsf{W})(x \leq t \wedge A), \\
(\forall x \leq t)A & := & (\forall x \in \mathsf{W})(x \leq t \to A), \\
(t : \mathsf{W} \to \mathsf{W}) & := & (\forall x \in \mathsf{W})(tx \in \mathsf{W}), \\
(t : \mathsf{W}^{m+1} \to \mathsf{W}) & := & (\forall x \in \mathsf{W})(tx : \mathsf{W}^m \to \mathsf{W}).
\end{array}$$

Before we turn to precise axiomatizations, let us give a short informal interpretation of the syntax of the language $\mathcal{L}$. The individual variables are conceived of as ranging over a universe V of computationally amenable objects, which can freely be applied to each other. Self-application is meaningful, but not necessarily total. V is assumed to be combinatory complete, due to the presence of the well-known combinators k and s, and V is closed under pairing. There is a collection of objects $W \subseteq V$, consisting of finite sequences of 0's and 1's. W is closed under various kinds of successor and predecessor operations as well as definition by cases. In addition, there are operations for the initial subword relation as well as the tally length of a binary word. Possibly, operations for word concatenation and/or word multiplication are explicitly included.

We now introduce the *basic theory of operations and words* B. The underlying logic of B is the *classical* logic of partial terms due to Beeson [3, 4]; it corresponds to E^+ logic with strictness and equality of Troelstra and Van Dalen [69]. According to this logic, quantifiers range over defined objects only, so that the usual axioms for $\exists$ and $\forall$ are modified to

$$A(t) \wedge t{\downarrow} \,\rightarrow\, (\exists x)A(x) \quad \text{and} \quad (\forall x)A(x) \wedge t{\downarrow} \,\rightarrow\, A(t),$$

and one further assumes that $(\forall x)(x{\downarrow})$. The *strictness axioms* claim that if a compound term is defined, then so also are all its subterms, and if a positive atomic statement holds, then all terms involved in that statement are defined. Note that $t{\downarrow} \leftrightarrow (\exists x)(t = x)$, so definedness need not be taken as basic symbol. The reader is referred to [3, 4, 69] for a detailed exposition of the logic of partial terms.

We are now ready to spell out in detail the *non-logical axioms* of B. To improve readability we divide the axioms into the following six groups.

I. Partial combinatory algebra and pairing

(1) $\mathsf{k}xy = x$,

(2) $\mathsf{s}xy{\downarrow} \,\wedge\, \mathsf{s}xyz \simeq xz(yz)$,

(3) $\mathsf{p}_0(x, y) = x \,\wedge\, \mathsf{p}_1(x, y) = y$.

II. Definition by cases on W

(4) $a \in \mathsf{W} \wedge b \in \mathsf{W} \wedge a = b \,\rightarrow\, \mathsf{d}_\mathsf{W}xyab = x$,

(5) $a \in \mathsf{W} \wedge b \in \mathsf{W} \wedge a \neq b \,\rightarrow\, \mathsf{d}_\mathsf{W}xyab = y$.

III. Closure, binary successors and predecessor

(6) $\epsilon \in \mathsf{W} \,\wedge\, (\forall x \in \mathsf{W})(\mathsf{s}_0 x \in \mathsf{W} \wedge \mathsf{s}_1 x \in \mathsf{W})$,

(7) $\mathsf{s}_0 x \neq \mathsf{s}_1 y \,\wedge\, \mathsf{s}_0 x \neq \epsilon \,\wedge\, \mathsf{s}_1 x \neq \epsilon$,

(8) $\mathsf{p}_\mathsf{W} : \mathsf{W} \to \mathsf{W} \,\wedge\, \mathsf{p}_\mathsf{W}\epsilon = \epsilon$,

(9) $x \in \mathsf{W} \,\to\, \mathsf{p}_\mathsf{W}(\mathsf{s}_0 x) = x \,\wedge\, \mathsf{p}_\mathsf{W}(\mathsf{s}_1 x) = x$,

(10) $x \in \mathsf{W} \wedge x \neq \epsilon \,\to\, \mathsf{s}_0(\mathsf{p}_\mathsf{W} x) = x \vee \mathsf{s}_1(\mathsf{p}_\mathsf{W} x) = x$.

IV. Lexicographic successor and predecessor

(11) $\mathsf{s}_\ell : \mathsf{W} \to \mathsf{W} \,\wedge\, \mathsf{s}_\ell\epsilon = 0$,

(12) $x \in \mathsf{W} \,\to\, \mathsf{s}_\ell(\mathsf{s}_0 x) = \mathsf{s}_1 x \,\wedge\, \mathsf{s}_\ell(\mathsf{s}_1 x) = \mathsf{s}_0(\mathsf{s}_\ell x)$,

(13) $\mathsf{p}_\ell : \mathsf{W} \to \mathsf{W} \,\wedge\, \mathsf{p}_\ell\epsilon = \epsilon$,

(14) $x \in \mathsf{W} \,\to\, \mathsf{p}_\ell(\mathsf{s}_\ell x) = x$,

(15) $x \in \mathsf{W} \wedge x \neq \epsilon \,\to\, \mathsf{s}_\ell(\mathsf{p}_\ell x) = x$.

V. Initial subword relation.

(16) $x \in \mathsf{W} \wedge y \in \mathsf{W} \,\to\, \mathsf{c}_\subseteq xy = 0 \vee \mathsf{c}_\subseteq xy = 1$,

(17) $x \in \mathsf{W} \,\to\, (x \subseteq \epsilon \leftrightarrow x = \epsilon)$,

(18) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \,\to\, (x \subseteq y \leftrightarrow x \subseteq \mathsf{p}_\mathsf{W} y \vee x = y)$,

(19) $x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge z \in \mathsf{W} \wedge x \subseteq y \wedge y \subseteq z \,\to\, x \subseteq z$.

VI. Tally length of binary words

(20) $\mathsf{l}_\mathsf{W} : \mathsf{W} \to \mathsf{W} \,\wedge\, \mathsf{l}_\mathsf{W}\epsilon = \epsilon$,

(21) $x \in \mathsf{W} \,\to\, \mathsf{l}_\mathsf{W}(\mathsf{s}_0 x) = \mathsf{s}_1(\mathsf{l}_\mathsf{W} x) \,\wedge\, \mathsf{l}_\mathsf{W}(\mathsf{s}_1 x) = \mathsf{s}_1(\mathsf{l}_\mathsf{W} x)$,

(22) $x \in \mathsf{W} \wedge \mathsf{l}_\mathsf{W}(x) = x \,\to\, \mathsf{l}_\mathsf{W}(\mathsf{s}_\ell x) = \mathsf{s}_1 x$,

(23) $x \in \mathsf{W} \wedge \mathsf{l}_\mathsf{W}(x) \neq x \,\to\, \mathsf{l}_\mathsf{W}(\mathsf{s}_\ell x) = \mathsf{l}_\mathsf{W}(x)$,

(24) $x \in \mathsf{W} \wedge y \in \mathsf{W} \,\to\, x \leq y \vee y \leq x$.

Let us immediately turn to two crucial consequences of the partial combinatory algebra axioms (1) and (2) of B, namely *abstraction* and *recursion*. These two central results appear in slightly different form than in the setting of a total combinatory algebra, the essential ingredients in the proofs, however, are the same. The relevant arguments are given, for example, in Beeson [3] or Feferman [24].

Lemma 2 (Abstraction) *For each $\mathcal{L}$ term t and all variables x there exists an $\mathcal{L}$ term $(\lambda x.t)$ whose variables are those of t, excluding x, so that* B *proves*

$$(\lambda x.t)\downarrow \ \wedge\ (\lambda x.t)x \simeq t.$$

As usual, we generalize λ abstraction to several arguments by iterating abstraction for one argument, i.e., $(\lambda x_1 \dots x_n.t)$ abbreviates $(\lambda x_1.(\dots(\lambda x_n.t)))$.

Lemma 3 (Recursion) *There exists a closed $\mathcal{L}$ term* rec *so that* B *proves*

$$\mathsf{rec} f\downarrow \ \wedge\ \mathsf{rec} f x \simeq f(\mathsf{rec} f)x.$$

Clearly, recursion nicely demonstrates the power of self-application. It will be an essential tool for defining operations in the various applicative systems to be introduced below.

In the meanwhile let us briefly sketch B's standard recursion-theoretic model *PRO* of *partial recursive operations.* The universe of *PRO* consists of the set of finite 0-1 sequences $\mathbb{W} = \{0,1\}^*$, and W is interpreted by $\mathbb{W}$. Application $\cdot$ is interpreted as partial recursive function application, i.e. $x \cdot y$ means $\{x\}(y)$ in *PRO*, where $\{x\}$ is a standard enumeration of the partial recursive functions over $\mathbb{W}$. It is easy to find interpretations of the constants of $\mathcal{L}$ so that all the axioms of B are true in *PRO*.

There are many more interesting models of the combinatory axioms, which can easily be extended to models of B. These include further recursion-theoretic models, term models, continuous models, generated models, and set-theoretic models. For detailed descriptions and results the reader is referred to Beeson [3], Feferman [26], and Troelstra and van Dalen [70]. We will make use of the so-called *extensional term model* of B in our upper bound arguments in Section 6; there we will define this model in some detail.

We finish this subsection by spelling out the obvious axioms for word concatenation and word multiplication in our applicative framework. Note, however, that these axioms do not belong to the theory B.

VII. Word concatenation.

(25) $* : \mathsf{W}^2 \to \mathsf{W}$,

(26) $x \in \mathsf{W} \ \to\ x{*}\epsilon = x$,

(27) $x \in \mathsf{W} \wedge y \in \mathsf{W} \ \to\ x{*}(\mathsf{s}_0 y) = \mathsf{s}_0(x{*}y) \ \wedge\ x{*}(\mathsf{s}_1 y) = \mathsf{s}_1(x{*}y)$.

VIII. Word multiplication.

(28) $\times : \mathsf{W}^2 \to \mathsf{W}$,

(29) $x \in \mathsf{W} \;\to\; x{\times}\epsilon = \epsilon$,

(30) $x \in \mathsf{W} \wedge y \in \mathsf{W} \;\to\; x{\times}(\mathsf{s}_0 y) = (x{\times}y){*}x \;\wedge\; x{\times}(\mathsf{s}_1 y) = (x{\times}y){*}x$.

In the following we write $\mathsf{B}(*)$ for the extension of B by the axioms (25)–(27), and $\mathsf{B}(*, \times)$ for B plus the axioms (25)–(30).

3.2 Bounded forms of induction

We have not yet specified induction principles on the binary words W; these are of course crucial for our proof-theoretic characterizations of complexity classes below. We start by defining three central classes of $\mathcal{L}$ formulas.

We call an $\mathcal{L}$ formula *positive* if it is built from the atomic formulas by means of disjunction, conjunction as well as existential and universal quantification over individuals; i.e., the positive formulas are exactly the implication and negation free $\mathcal{L}$ formulas. We let Pos stand for the collection of positive formulas. Further, an $\mathcal{L}$ formula is called W *free*, if the relation symbol W does not occur in it.

Most important in the sequel are the so-called *bounded (with respect to* W*) existential formulas* or $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ *formulas* of $\mathcal{L}$. A formula $A(f, x)$ belongs to the class $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ if it has the form $(\exists y \leq fx)B(f, x, y)$ for $B(f, x, y)$ a *positive and* W *free* formula. It is important to recall here that bounded quantifiers range over W, i.e., $(\exists y \leq fx)B(f, x, y)$ stands for

$$(\exists y \in \mathsf{W})[y \leq fx \;\wedge\; B(f, x, y)].$$

Further observe that the matrix B of a $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula can have unrestricted existential and universal individual quantifiers, not ranging over W, however.

Assuming that the bounding operation f in a $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula has polynomial growth, $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formulas can be seen as a very abstract applicative analogue of Buss' Σ^b_1 formulas (cf. [8]) or Ferreira's NP formulas (cf. [32, 33]). Notice, however, whereas the latter classes of formulas define exactly the NP predicates, $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formulas of $\mathcal{L}$ in general define highly undecidable sets in the standard recursion theoretic model PRO.

At the heart of our delineation of complexity classes below are forms of bounded (with respect to W) induction. These principles allow induction with respect to formulas in the class $\Sigma^{\mathsf{b}}_{\mathsf{W}}$, under the proviso that the bounding operation f has the right type. We will distinguish usual notation induction on binary words and the corresponding "slow" induction principle with respect to the lexicographic successor s_ℓ.

The scheme $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ of $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ *notation induction on* W includes for each formula $A(x) \equiv (\exists y \leq fx)B(f, x, y)$ in the formula class $\Sigma^{\mathsf{b}}_{\mathsf{W}}$,

$$(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}}) \qquad \begin{aligned} f : \mathsf{W} \to \mathsf{W} \,\wedge\, A(\epsilon) \,\wedge\, (\forall x \in \mathsf{W})(A(x) \to A(\mathsf{s}_0 x) \wedge A(\mathsf{s}_1 x)) \\ \to (\forall x \in \mathsf{W})A(x) \end{aligned}$$

Accordingly, the induction scheme $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$ of $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ *lexicographic induction on* W claims for each formula $A(x) \equiv (\exists y \leq fx)B(f, x, y)$ in the class $\Sigma^{\mathsf{b}}_{\mathsf{W}}$,

$$(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell}) \qquad \begin{aligned} f : \mathsf{W} \to \mathsf{W} \,\wedge\, A(\epsilon) \,\wedge\, (\forall x \in \mathsf{W})(A(x) \to A(\mathsf{s}_{\ell} x)) \\ \to (\forall x \in \mathsf{W})A(x) \end{aligned}$$

Let us mention that notation induction I_{W} and lexicographic induction I_{ℓ} correspond to what is usually called PIND (equivalently: LIND) and IND, respectively, in the classical literature on bounded arithmetic, cf. e.g. Buss [8] or Beckmann [2]. However, we will see that the *strength* of our induction principles in the applicative setting may differ from the corresponding formal setting in bounded arithmetic: for example, the theory $\mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$, termed PS below, at first sight seems to resemble Buss' theory T^1_2, but as we will prove in the course of this paper, the provably total functions of the theory $\mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$ are all polynomial space computable functions. This difference in strength is due to the extremely strong expressive power of our applicative systems.

We will prove in the next section (Lemma 6) that indeed $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$ entails $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ over our base theory B. This is similar to the fact that Buss' T^1_2 is an extension of S^1_2. Further, let us mention that the principles of set induction and NP induction considered in Strahm [62] (cf. also Cantini [17]) are directly entailed by $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$. Moreover, also the axiom of operation induction of Jäger and Strahm [46] is covered by the above bounded induction schemes. An induction principle related to $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ has previously been studied by Cantini [13] in the context of polynomially bounded operations (cf. also Cantini [17]).

Depending on whether we include $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ or $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$, and whether we assume as given only word concatenation or both word concatenation and word multiplication, we can now distinguish the following four applicative theories PT, PTLS, PS, and LS:

$$\begin{aligned} \mathsf{PT} &:= \mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}}) \qquad & \mathsf{PTLS} &:= \mathsf{B}(*) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}}) \\ \mathsf{PS} &:= \mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell}) \qquad & \mathsf{LS} &:= \mathsf{B}(*) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell}) \end{aligned}$$

As the naming of these system suggests, it is our aim in the sequel to establish that the provably total operations on words of PT, PTLS, PS, and LS coincide with FPtime, FPtimeLinspace, FPspace, and FLinspace, respectively. On our way we will also be interested in some higher type aspects of our applicative systems.

4 Deriving bounded recursions

It is the main purpose of this section to show that the provably total word functions of the systems PT, PTLS, PS, and LS include the classes FPtime, FPtimeLinspace, FPspace, and FLinspace, respectively. We set up our lower bound arguments in such a way as to facilitate the discussion on higher type issues in the subsequent section.

Let us first start with a formal definition of the notion of *provably total function* of a given $\mathcal{L}$ theory. First note that for each word $w \in \mathbb{W}$ we have a canonical closed term $\overline{w}$ of $\mathcal{L}$ which represents w; of course, $\overline{w}$ is constructed form ϵ by means of the successor operations s_0 and s_1. In the sequel we sometimes identify $\overline{w}$ with w when working in the language $\mathcal{L}$. A function $F : \mathbb{W}^n \to \mathbb{W}$ is called *provably total in an $\mathcal{L}$ theory* T, if there exists a closed $\mathcal{L}$ term t_F such that

(i) $\mathsf{T} \vdash t_F : \mathsf{W}^n \to \mathsf{W}$ and, in addition,

(ii) $\mathsf{T} \vdash t_F \overline{w}_1 \cdots \overline{w}_n = \overline{F(w_1, \dots, w_n)}$ for all $w_1, \dots, w_n$ in $\mathbb{W}$.

The notion of a provably total word function is divided into two conditions (i) and (ii). The first condition (i) expresses that t_F is a total operation from W^n to W, *provably in the $\mathcal{L}$ theory* T. Condition (ii), on the other hand, claims that t_F indeed represents the given function $F : \mathbb{W}^n \to \mathbb{W}$, for each fixed word w in $\mathbb{W}$.

Observe that one gets a too weak notion of provably total function if one drops condition (i). For example, in the theory B it is well-known that one can represent *all recursive functions* in the sense of (ii). The proof of this fact runs completely analogous to the argument in the untyped λ calculus showing that all recursive function are representable there (cf. [1, 38]). The crucial ingredient in the proof is of course the recursion or fixed point lemma (Lemma 3). Hence, for example, it is possible to find a closed $\mathcal{L}$ term exp representing a suitable form of exponentiation on $\mathbb{W}$ in the sense of condition (ii) above, but indeed none of the theories introduced in the previous section is able to derive the totality or convergence statement $\mathsf{exp} : \mathsf{W} \to \mathsf{W}$.

Our general strategy for proving lower bounds in the sequel is to make use of the function algebra characterizations of our complexity classes which we have discussed in Section 2. Crucial in the set up of the four function algebras of Theorem 1 are two forms of bounded recursion, namely *bounded recursion on notation* (BRN) and *bounded lexicographic recursion* (BRL). We will now show that (BRN) and (BRL) can be very smoothly and naturally represented in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ and $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$, respectively. The key in the proof below is the recursion or fixed point theorem (Lemma 3) and of course our carefully chosen forms of bounded induction.

In the sequel we also need the cut-off operator $|$ in order to describe bounded recursion in our systems. Informally speaking, $t \mid s$ is t if $t \le s$ and s else. More formally, we can make use of definition by cases $\mathsf{d_W}$ and the characteristic function $\mathsf{c}_\subseteq$ in order to define $|$; then $t \mid s$ simply is an abbreviation for the $\mathcal{L}$ term $\mathsf{d_W} ts(\mathsf{c}_\subseteq(\mathsf{l_W} t)(\mathsf{l_W} s))0$.

Let us now first turn to bounded recursion on notation (BRN) in the system $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$. In favor of a more compact and uniform presentation we state this form of recursion in our applicative setting by making use of one step function only and using the predecessor operation $\mathsf{p_W}$ instead. Moreover, in order to simplify notation, we have only displayed one parameter; the general case with an arbitrary list of parameters is completely analogous.

Lemma 4 *There exists a closed $\mathcal{L}$ term $\mathsf{r_W}$ so that $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ proves*

$$f : \mathsf{W} \to \mathsf{W} \land g : \mathsf{W}^3 \to \mathsf{W} \land b : \mathsf{W}^2 \to \mathsf{W} \;\to$$

$$\begin{cases} \mathsf{r_W} fgb : \mathsf{W}^2 \to \mathsf{W} \land \\ x \in \mathsf{W} \land y \in \mathsf{W} \land y \neq \epsilon \land h = \mathsf{r_W} fgb \;\to \\ \qquad hx\epsilon = fx \land hxy = gxy(hx(\mathsf{p_W} y)) \mid bxy \end{cases}$$

Proof The crucial strategy of this proof consists in applying the recursion or fixed point lemma (Lemma 3) in order to define the term $\mathsf{r_W}$ and make subsequent use of $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ in order to establish the required totality or convergence assertion about $\mathsf{r_W}$.

We first define t to be the following $\mathcal{L}$ term depending on f, g, and b,

$$t \;:=\; \lambda hxy.\mathsf{d_W} f(\lambda z.gzy(hz(\mathsf{p_W} y)) \mid bzy)\epsilon yx,$$

and then set $\mathsf{r_W} \;:=\; \lambda fgb.\mathsf{rec}\, t$. We now have for $h \simeq \mathsf{r_W} fgb$,

$$hxy \simeq \mathsf{rec}\, txy \simeq t(\mathsf{rec}\, t)xy \simeq thxy \simeq \mathsf{d_W} f(\lambda z.gzy(hz(\mathsf{p_W} y)) \mid bzy)\epsilon yx.$$

In particular, we obtain for all x and y in W with $y \neq \epsilon$,

(1) $$hx\epsilon \simeq fx \ \wedge \ hxy \simeq gxy(hx(\mathsf{p_W}y)) \mid bxy.$$

In the following we reason in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$ and assume in addition that

(2) $$f : \mathsf{W} \to \mathsf{W} \ \wedge \ g : \mathsf{W}^3 \to \mathsf{W} \ \wedge \ b : \mathsf{W}^2 \to \mathsf{W}.$$

Our crucial task is to show that indeed $h : \mathsf{W}^2 \to \mathsf{W}$, and this is of course where bounded induction enters the scene. First, let c be an operation so that cxy is simply fx if $y = \epsilon$ and bxy, otherwise. Obviously, we have that $c : \mathsf{W}^2 \to \mathsf{W}$. Now we define $A(y)$ to be the $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula

$$A(y) \ := \ (\exists z \leq cxy)(hxy = z).$$

Recall at this point that bounded quantifiers range over W. Fixing the parameter $x \in \mathsf{W}$, it is now a matter of routine to derive from (1) and (2),

(3) $$A(\epsilon) \ \wedge \ (\forall y \in \mathsf{W})(A(y) \ \to \ A(\mathsf{s_0}y) \wedge A(\mathsf{s_1}y)).$$

Further, (3) brings us in the position to apply notation induction for $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formulas, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$, and we can thus conclude

(4) $$(\forall y \in \mathsf{W})(\exists z \in \mathsf{W})(z \leq cxy \wedge hxy = z),$$

for an arbitrarily chosen x in W. But (4) shows indeed that we have established h to be an operation from W^2 to W, i.e., $h : \mathsf{W}^2 \to \mathsf{W}$. This is as claimed and ends our proof. $\square$

We want to emphasize that indeed we have established the existence of a type two functional for bounded recursion on notation in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$; this will be the key for interpreting the Cook-Urquhart system PV^{ω} into PT in the next section. At any rate, the previous lemma shows that the functions in FPTIME and FPTIMELINSPACE are provably total in PT and PTLS, respectively. Moreover, observe that in fact we have only used very special instances of $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$, namely $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$ has been applied for statements of the form $(\exists z \leq fy)(gy = z)$.

If we replace notation induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$, by lexicographic induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$, then of course one expects that we can derive bounded lexicographic recursion (BRL) instead of bounded recursion on notation (BRN). The proof of this fact runs completely analogous to the proof of the previous lemma and is hence omitted. Clearly, the following lemma shows that FPSPACE and FLINSPACE are contained in the provably total functions of PS and LS, respectively.

Lemma 5 *There exists a closed $\mathcal{L}$ term r_ℓ so that* $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ *proves*

$$f : \mathsf{W} \to \mathsf{W} \wedge g : \mathsf{W}^3 \to \mathsf{W} \ \wedge \ b : \mathsf{W}^2 \to \mathsf{W} \ \ \to$$

$$\begin{cases} \mathsf{r}_\ell fgb : \mathsf{W}^2 \to \mathsf{W} \wedge \\ x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \wedge h = \mathsf{r}_\ell fgb \ \to \\ \qquad hx\epsilon = fx \wedge hxy = gxy(hx(\mathsf{p}_\ell y)) \mid bxy \end{cases}$$

A natural question to ask is whether bounded recursion on notation in the above functional form using the recursor r_W is directly available in $\mathsf{B}+(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$, too. The answer is indeed positive due to the fact that over the base theory B, lexicographic induction $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ entails notation induction $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\mathsf{W})$. The proof of this fact is completely analogous to the argument showing that Buss' theory T^1_2 contains his system S^1_2, cf. Buss [8]. Nevertheless, since our setting is different, and in some sense simpler, we spell out the relevant arguments in some detail.

Lemma 6 *We have that* $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ *entails* $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\mathsf{W})$ *over our base theory* B.

Proof Let us work informally in the theory $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$. By the previous lemma, bounded lexicographic recursion is at our disposal. Hence, we can define the well-known "most significant part" function $\mathsf{msp} : \mathsf{W}^2 \to \mathsf{W}$,

$$\mathsf{msp}a\epsilon = a, \quad \mathsf{msp}ab = \mathsf{p}_\mathsf{W}(\mathsf{msp}a(\mathsf{p}_\ell b)), \quad \mathsf{msp}ab \leq a,$$

for all a in W and b in W with $b \neq \epsilon$. This function cuts off b bits to the right of a, where b is understood in the sense of the lexicographic ordering $<_\ell$ on W. Further, we have the cut-off operation $\dot{-} : \mathsf{W}^2 \to \mathsf{W}$,

$$a \dot{-} \epsilon = a, \quad a \dot{-} b = \mathsf{p}_\ell(a \dot{-} \mathsf{p}_\ell b), \quad a \dot{-} b \leq a,$$

for $a, b \in \mathsf{W}$ and $b \neq \epsilon$. Finally, the length function $|\cdot| : \mathsf{W} \to \mathsf{W}$, which measures the length of a word by means of the $<_\ell$ ordering can be defined by bounded lexicographic recursion and by making us of the "tally" length function which is available in B,

$$|\epsilon| = \epsilon, \quad |a| = \mathsf{if}\ \mathsf{p}_\ell a < a \ \mathsf{then}\ \mathsf{s}_\ell|\mathsf{p}_\ell a| \ \mathsf{else}\ |\mathsf{p}_\ell a|, \quad |a| \leq a,$$

for all a in W with $a \neq \epsilon$. It is not difficult to see that the usual properties of msp, $\dot{-}$, and $|\cdot|$ are derivable in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$.

Consider now an arbitrary $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula $A(x) \equiv (\exists y \leq fx)B(f, x, y)$ and assume the premise of $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\mathsf{W})$, i.e.,

(1) $\qquad f : \mathsf{W} \to \mathsf{W} \ \wedge \ A(\epsilon) \ \wedge \ (\forall x \in \mathsf{W})(A(x) \to A(\mathsf{s}_0 x) \wedge A(\mathsf{s}_1 x)).$

We fix an $a \in \mathsf{W}$ and aim at showing $A(a)$. For that purpose we let $C(a,x)$ be the formula $A(\mathsf{msp}a(|a| \dot{-} x))$. Observe that since $f : \mathsf{W} \to \mathsf{W}$, we also have $g : \mathsf{W} \to \mathsf{W}$, for g being the operation $\lambda x.f(\mathsf{msp}a(|a| \dot{-} x))$. We can now readily derive from (1),

$$(2) \qquad g : \mathsf{W} \to \mathsf{W} \wedge C(a,\epsilon) \wedge (\forall x \in \mathsf{W})(C(a,x) \to C(a,\mathsf{s}_\ell x)).$$

This brings us in the position to apply $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ in order to derive the statement $(\forall x \in \mathsf{W})C(a,x)$ and, in particular, $C(a,|a|)$. Clearly, $A(a)$ is entailed by $C(a,|a|)$. We have established in $\mathsf{B}+(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ the schema of notation induction on W for $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formulas, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$. □

Corollary 7 *The assertion of Lemma 4 is derivable in* $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$.

Corollary 8 *We have that* PT *and* PTLS *are directly contained in* PS *and* LS*, respectively.*

In this section we have established lower bounds in terms of provably total functions of the four central systems, PT, PTLS, PS, and LS. We collect the corresponding results in the following theorem.

Theorem 9 *We have the following lower bound results:*

1. *The provably total functions of* PT *include* FPtime.
2. *The provably total functions of* PTLS *include* FPtimeLinspace.
3. *The provably total functions of* PS *include* FPspace.
4. *The provably total functions of* LS *include* FLinspace.

Finally, let us mention that the results of this section entail that the applicative theories PTO and PTO^+ introduced and analyzed in Strahm [62] are directly contained in our system PT. In particular, the induction principles presented in [62] directly follow from the more general induction principle $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$, and the axioms about bounded recursion on notation in [62] are derivable in PT thanks to Lemma 4.

5 Higher types in PT and the system PV^ω

In the last decade intense research efforts have been made in the area of so-called higher type complexity theory and, in particular, feasible functionals of higher types. This research is still ongoing and it is not yet clear what the right higher type analogue of the polynomial time computable functions is. Most prominent in the previous research is the class of so-called *basic feasible functionals* BFF, which has proved to be a very robust class with various kinds of interesting characterizations.

The basic feasible functionals of type 2, BFF_2, were first studied in Melhorn [53]. More than ten years later in 1989, Cook and Urquhart [23] introduced the basic feasible functionals at all finite types in order to provide functional interpretations of feasibly constructive arithmetic; in particular, they defined a typed formal system PV^ω and used it to establish functional and realizability interpretations of an intuitionistic version of Buss' theory S^1_2. The basic feasible functionals BFF are exactly those functionals which can be defined by PV^ω terms. Subsequently, much work has been devoted to BFF, cf. e.g. Cook and Kapron [22, 47], Irwin, Kapron and Royer [41], Pezzoli [55], Royer [57], and Seth [59].

In this section we introduce an intensional and an extensional version of the Cook-Urquhart system PV^ω and show that both systems are naturally contained in our applicative system PT. Hence, in a sense, PT *provides a direct justification of* PV^ω *in a type-free applicative setting.* In addition, the embeddings established in the sequel also show that the well-known systems of bounded arithmetic PTCA and PTCA^+ of Ferreira [32, 33] or, equivalently, Cook's system PV [21] and Buss' S^1_2 [8] are directly contained in PT.

We start off with defining the collection $\mathcal{T}$ *of finite type symbols* $(\alpha, \beta, \gamma, \ldots)$. $\mathcal{T}$ is inductively generated by the usual clauses, (i) $0 \in \mathcal{T}$, (ii) if $\alpha, \beta \in \mathcal{T}$, then $(\alpha \times \beta) \in \mathcal{T}$, and (iii) if $\alpha, \beta \in \mathcal{T}$, then $(\alpha \to \beta) \in \mathcal{T}$. Hence, we have product and function types as usual. Observe, however, that in our setting the ground type 0 stands for the set of binary words and not for the set of natural numbers. We use the usual convention and write $\alpha_1 \to \alpha_2 \to \cdots \to \alpha_k$ instead of $(\alpha_1 \to (\alpha_2 \to \cdots \to (\alpha_{k-1} \to \alpha_k) \cdots))$.

In the following we sketch a version of PV^ω which is similar in spirit to the presentation of Heyting's arithmetic in all finite types HA^ω in Troelstra and Van Dalen [70]; however, the logic of PV^ω is classical logic. PV^ω is based on combinators and noncommittal as to the exact nature of equality between objects of higher types. Later we will also discuss an extensional version EPV^ω of PV^ω.[1]

The language of PV^ω includes for each type symbol $\alpha \in \mathcal{T}$ a countable

[1]Actually, the system EPV^ω introduced below corresponds to the Cook-Urquhart system IPV^ω in [23] *with classical logic* instead of intuitionistic logic. What we call PV^ω in this paper is just an intensional version of EPV^ω. We follow Troelstra and Van Dalen [70] in using this terminology.

collection $x^\alpha, y^\alpha, z^\alpha, u^\alpha, v^\alpha, w^\alpha, \dots$ of variables of type α. Further, for each $\alpha \in \mathcal{T}$ we have a binary relation symbol $=^\alpha$ for equality at type α, and for all $\alpha, \beta \in \mathcal{T}$ there is an application operator $\cdot^{\alpha,\beta}$. The *constants* of PV^ω first of all include the "arithmetical" constants of $\mathcal{L}$, namely $\epsilon, \mathsf{s}_0, \mathsf{s}_1, \mathsf{p}_\mathsf{W}, \mathsf{s}_\ell, \mathsf{p}_\ell, \mathsf{c}_\subseteq, \mathsf{l}_\mathsf{W}, *$, and $\times$; these constants now receive their obvious types in the typed language of PV^ω. In addition, we have typed versions of $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1$ as well as d_W, and most importantly, a recursion operator r. More precisely, we have for all types $\alpha, \beta, \gamma \in \mathcal{T}$ the following constants with their associated types:

$$\begin{aligned}
\mathsf{k}^{\alpha,\beta} &: \alpha \to \beta \to \alpha, \\
\mathsf{s}^{\alpha,\beta,\gamma} &: (\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma, \\
\mathsf{p}^{\alpha,\beta} &: \alpha \to \beta \to (\alpha \times \beta), \\
\mathsf{p}_0^{\alpha,\beta} &: (\alpha \times \beta) \to \alpha, \\
\mathsf{p}_1^{\alpha,\beta} &: (\alpha \times \beta) \to \beta, \\
\mathsf{d}^{\alpha} &: \alpha \to \alpha \to 0 \to 0 \to \alpha, \\
\mathsf{r} &: 0 \to (0 \to 0 \to 0) \to (0 \to 0) \to 0 \to 0.
\end{aligned}$$

In the sequel we often omit the type superscripts of variables and constants if these are clear from the context or unimportant.

The *terms of* PV^ω are now generated from the variables and constants by the expected clause for application, namely: if t is a term of type $(\alpha \to \beta)$ and s a term of type α, then $(t \cdot^{\alpha,\beta} s)$ is a term of type β. As usual we write (ts) instead of $(t \cdot^{\alpha,\beta} s)$; moreover, outer parenthesis are often dropped, and we make free use of the convention of association to the left when writing applicative terms. The *formulas of* PV^ω are built from the prime formulas $(t =^\alpha s)$ for t, s of type α, by means of $\neg$, $\wedge$, $\vee$, $\to$, $(\forall x^\alpha)$, and $(\exists x^\alpha)$. As in the applicative setting above, we call a formula *positive*, if it is implication and negation free.

The logic of PV^ω is many-sorted *classical* predicate calculus with equality. The non-logical axioms of PV^ω include the defining axioms for the constants of PV^ω: these consist of (i) the defining axioms for the "arithmetical" constants of PV^ω, which are just the obvious rewriting to the typed setting of the corresponding axioms of B, and (ii) the following axioms for the combinators $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1$ and r:

$$\begin{array}{lll}
\mathsf{k}xy = x, & \mathsf{s}xyz = xz(yz), & \\
\mathsf{p}_0(\mathsf{p}xy) = x, & \mathsf{p}_1(\mathsf{p}xy) = y, & \mathsf{p}(\mathsf{p}_0 z)(\mathsf{p}_1 z) = z, \\
\mathsf{d}xyuu = x, & u \neq v \to \mathsf{d}xyuv = y, & \\
\mathsf{r}xyz\epsilon = x, & u \neq \epsilon \to \mathsf{r}xyzu = yu(\mathsf{r}xyz(\mathsf{p}_\mathsf{W}u)) \mid zu. &
\end{array}$$

In the defining equations for r, the cut-off operator $|$ is understood in the same way as in the untyped applicative setting via the definition by cases operator d and the characteristic function $\mathsf{c}_{\subseteq}$. We have that r provides a type two functional for bounded recursion on notation in the natural expected manner. Finally, the system PV^{ω} includes induction on notation,

$$A(\epsilon) \wedge (\forall x^0)(A(x) \rightarrow A(\mathsf{s}_0 x) \wedge A(\mathsf{s}_1 x)) \;\rightarrow\; (\forall x^0)A(x),$$

for all formulas $A(x)$ in the language of the system PV^{ω} which have the shape $(\exists y \leq tx)B(x,y)$, with B being a positive and quantifier free formula and t a term of type $(0 \rightarrow 0)$.

As usual, the availability of the typed combinators k and s allows for the definition of simply typed λ terms $(\lambda x^{\alpha}.t)$, for each type symbol $\alpha \in \mathcal{T}$. The definition follows the usual pattern, cf. e.g. [70].

Let us mention once more that in PV^{ω} we do not claim that equality $=^{\alpha}$ for α a higher type is extensional equality. Accordingly, we now sketch an embedding of PV^{ω} into PT by means of the abstract *intensional type structure* $\langle(\mathsf{IT}_{\alpha}, =)\rangle_{\alpha\in\mathcal{T}}$. This embedding is analogous to the embedding of HA^{ω} into the theory of operations and numbers APP in [70]. We work in the applicative language $\mathcal{L}$ and define IT_{α} inductively as follows:

$$\begin{aligned}
x \in \mathsf{IT}_0 \quad &:= \quad x \in \mathsf{W},\\
x \in \mathsf{IT}_{\alpha\times\beta} \quad &:= \quad \mathsf{p}_0 x \in \mathsf{IT}_{\alpha} \;\wedge\; \mathsf{p}_1 x \in \mathsf{IT}_{\beta} \;\wedge\; \mathsf{p}(\mathsf{p}_0 x)(\mathsf{p}_1 x) = x,\\
x \in \mathsf{IT}_{\alpha\rightarrow\beta} \quad &:= \quad (\forall y \in \mathsf{IT}_{\alpha})(xy \in \mathsf{IT}_{\beta}).
\end{aligned}$$

Equality in IT_{α} is simply the restriction of equality in PT. We now get an embedding $(\cdot)^{\mathsf{IT}}$ of PV^{ω} into PT by letting the variables of type α range over IT_{α}. Further, application $\cdot^{\alpha,\beta}$ in PV^{ω} carries over to application $\cdot$ in PT, restricted to $\mathsf{IT}_{\alpha\rightarrow\beta}\times\mathsf{IT}_{\alpha}$. Moreover, the constants of PV^{ω} different from r are interpreted by the corresponding constants in $\mathcal{L}$. The recursor r of PV^{ω} can be interpreted, for example, by the closed $\mathcal{L}$ term $\lambda xyzu.\mathsf{r}_{\mathsf{W}}(\mathsf{k}x)(\mathsf{k}y)(\mathsf{k}z)\epsilon u$, where r_{W} denotes the closed term stated in the assertion of Lemma 4. We now have the following embedding theorem.

Theorem 10 *We have for all sentences A that $\mathsf{PV}^{\omega} \vdash A$ entails $\mathsf{PT} \vdash A^{\mathsf{IT}}$.*

Proof The proof of the theorem is immediate except for the case of recursion and induction in PV^{ω}. But the defining axioms for r and, more importantly, the fact the r has the right type, are readily derivable in PT by the results of Lemma 4. Moreover, the translation of notation induction in PV^{ω} directly carries over to $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ in PT; for, a formula of the form $(\exists y \leq tx)B(x,y)$

with t of type $(0 \to 0)$ and B positive and quantifier free directly translates into a $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula in the untyped applicative setting of PT. $\square$

In a further step we now turn to an extensional version EPV^ω of PV^ω; we also give an embedding of EPV^ω into our type-free applicative setting PT, which is analogous to the embedding of an extensional version EHA^ω of HA^ω into APP in [70]. The *extensionality axioms* $(\mathsf{Ext}_{\alpha,\beta})$ for all $\alpha, \beta \in \mathcal{T}$ are given in the expected manner by

$$(\mathsf{Ext}_{\alpha,\beta}) \qquad\qquad (\forall y, z)[(\forall x)(yx = zx) \to y = z],$$

for y, z of type $(\alpha \to \beta)$ and x of type α. Now EPV^ω is defined in the same way as PV^ω, except that (i) it includes $(\mathsf{Ext}_{\alpha,\beta})$ for all $\alpha, \beta \in \mathcal{T}$, and (ii) the induction formulas $(\exists y \le tx)B(x, y)$ of PV^ω are restricted in EPV^ω to positive quantifier free formulas B *not containing equalities of higher type.*

In our embedding of EPV^ω into PT we now make use of an abstract *extensional type structure* $\langle(\mathsf{ET}_\alpha, =_\alpha)\rangle_{\alpha\in\mathcal{T}}$ in $\mathcal{L}$, cf. [70]. ET_α and $=_\alpha$ are inductively given in the following manner:

$$\begin{array}{lcl}
x \in \mathsf{ET}_0 & := & x \in \mathsf{W}, \\
x =_0 y & := & x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge x = y, \\
x \in \mathsf{ET}_{\alpha\times\beta} & := & \mathsf{p}_0 x \in \mathsf{ET}_\alpha \wedge \mathsf{p}_1 x \in \mathsf{ET}_\beta \wedge \mathsf{p}(\mathsf{p}_0 x)(\mathsf{p}_1 x) = x, \\
x =_{\alpha\times\beta} y & := & (\mathsf{p}_0 x =_\alpha \mathsf{p}_0 y) \wedge (\mathsf{p}_1 x =_\alpha \mathsf{p}_1 y), \\
x \in \mathsf{ET}_{\alpha\to\beta} & := & (\forall y, z)(y =_\alpha z \to xy =_\beta xz), \\
x =_{\alpha\to\beta} y & := & x \in \mathsf{ET}_{\alpha\to\beta} \wedge y \in \mathsf{ET}_{\alpha\to\beta} \wedge (\forall z \in \mathsf{ET}_\alpha)(xz =_\beta yz).
\end{array}$$

EPV^ω can now be interpreted into PT via an embedding $(\cdot)^{\mathsf{ET}}$ in the same way as we have embedded PV^ω into PT via $(\cdot)^{\mathsf{IT}}$ above. Therefore, we omit the proof of the following theorem.

Theorem 11 *We have for all sentences A that $\mathsf{EPV}^\omega \vdash A$ entails $\mathsf{PT} \vdash A^{\mathsf{ET}}$.*

Let us observe that if we interpret PT in its standard recursion-theoretic model of partial recursive operations *PRO*, then IT and ET correspond to the so-called *hereditarily recursive operations HRO* and *hereditarily effective operations HEO*, respectively, cf. [70]. *HRO* forms the standard recursion-theoretic model of PV^ω and *HEO* is the corresponding interpretation of EPV^ω.

We finish this section with the observation that the results of the previous section give rise immediately to higher type systems for FPtimeLinspace, FPspace, and FLinspace, which are naturally contained in the corresponding type-free settings PTLS, PS and LS, respectively. For example, the type system for FPSPACE has a type two recursor for bounded lexicographic recursion, which is available in PS by Lemma 5. It might be of interest to study these type systems from the recursion-theoretic and abstract machine point of view.

6 Realizing positive derivations

It is the aim of this section to establish proof-theoretic upper bounds of PT, PTLS, PS, and LS. Namely, we will show that the lower bounds with respect to provably total functions derived in Theorem 9 are indeed sharp. For our upper bound arguments we will proceed in two steps. First, a *partial cut elimination* argument in a sequent-style reformulation of our four systems is employed in order to show that as far as the computational content of our systems is concerned, we can restrict ourselves to positive derivations, i.e., sequent style proofs using positive formulas only. In a second crucial step we use a notion of *realizability for positive formulas* in the standard open term model of our systems: quasi cut-free positive sequent derivations of PT, PTLS, PS, and LS are suitably realized by word functions in FPTIME, FPTIMELINSPACE, FPSPACE, and FLINSPACE, respectively, thus yielding the desired computational information concerning the provably total functions of these systems.

Actually, in the following we will establish our upper bounds for slight strengthenings of PT, PTLS, PS, and LS. Namely, we augment our applicative frameworks by the axioms (Tot) for *totality of application* and (Ext) for *extensionality of operations*,

$$(\mathsf{Tot}) \qquad (\forall x, y)(xy\downarrow) \qquad\qquad (\mathsf{Ext}) \qquad (\forall f, g)[(\forall x)(fx = gx) \to f = g]$$

We observe that $\mathsf{B} + (\mathsf{Tot})$ proves $t\downarrow$ for each term t, so that in the presence of (Tot) the logic of partial terms reduces to ordinary classical predicate calculus. Accordingly, if T denotes one of the systems PT, PTLS, PS, or LS, then we write T^+ for the system T based on ordinary classical logic with equality and augmented with the axiom of extensionality (Ext). Observe that in the setting of T^+ we no longer have the relation symbol $\downarrow$, so that instead of axiom (2) of B we simply have the usual total version of the s combinator, given by the axiom $\mathsf{s}xyz = xz(yz)$.

The simplest model of T^+ is just the standard open term model $\mathcal{M}(\lambda\eta)$, which is based on a straightforward extension of usual $\lambda\eta$ reduction. We will discuss this model is some more detail below, where it will be used in our realizability interpretation of (the positive fragment of) T^+.

The fact that the presence of (Tot) and (Ext) does not raise the strength of a given partial applicative system is not too surprising as is witnessed by the previous work on applicative theories. For sample references cf. Cantini [14, 17] and Jäger and Strahm [45].

6.1 Preparatory partial cut elimination

In this subsection we turn to a preparatory partial cut elimination argument for T^+, where again T denotes any of the systems PT, PTLS, PS, or LS. For that purpose, we will make use of a reformulation of T^+ in terms of Gentzen's classical sequent calculus LK; in the sequel we assume that the reader is familiar with LK as it is presented, for example, in Girard [35].

In the following we let $\Gamma, \Delta, \Lambda, \ldots$ range over finite *sequences* of formulas in the language $\mathcal{L}$; a *sequent* is a formal expression of the form $\Gamma \Rightarrow \Delta$. As usual, the natural interpretation of the sequent $A_1, \ldots, A_n \Rightarrow B_1, \ldots, B_m$ is $(A_1 \wedge \cdots \wedge A_n) \rightarrow (B_1 \vee \cdots \vee B_m)$.

We are now aiming at a suitable sequent-style reformulation of T^+. As mentioned above, our crucial aim is to prove a partial cut elimination theorem so that the only cuts occurring in partially cut free derivations have *positive cut formulas*. Hence, in order to solve this task, we must find a Gentzen-style reformulation of T^+ so that all the main formulas of non-logical axioms and rules (including equality) are positive. In the following we sketch such a reformulation of T^+; we are confining ourselves to the essential points without spelling out each single axiom and rule in detail.

The axioms of our basic theory of operations and words, B, are easily reformulated in positive form. Just to give an example, axioms (4) and (5) about definition by cases $\mathsf{d_W}$ on W translate into the pair of sequents

$$\mathsf{W}(r),\, \mathsf{W}(s),\, r = s \;\Rightarrow\; \mathsf{d_W} t_1 t_2 r s = t_1,$$
$$\mathsf{W}(r),\, \mathsf{W}(s) \;\Rightarrow\; r = s,\, \mathsf{d_W} t_1 t_2 r s = t_2,$$

for all terms r, s, t_1, t_2 of $\mathcal{L}$. Observe that as usual in sequent formulations, we take all substitution instances of the axioms of B. It is a matter of routine to spell out in positive sequent form the other axioms of B. In some cases, an axiom has to be split into several sequents, e.g., axiom (18) about the initial subword relation is now given by the two sequents

$$\mathsf{W}(s),\, \mathsf{W}(t),\, s \subseteq t \;\Rightarrow\; t = \epsilon,\, s \subseteq \mathsf{p_W} t,\, s = t,$$
$$\mathsf{W}(s),\, \mathsf{W}(t),\, s \subseteq \mathsf{p_W} t \vee s = t \;\Rightarrow\; t = \epsilon,\, s \subseteq t.$$

We leave it to the reader to provide suitable positive sequents of the other axioms of B, and also of the axioms (25)–(30) concerning word concatenation and word multiplication. Moreover, the extensionality axiom (Ext) of T^+ now simply takes the positive sequent form

$$(\forall x)(sx = tx) \;\Rightarrow\; s = t,$$

for s and t being arbitrary terms in our applicative language $\mathcal{L}$, not containing the variable x. Of course, T^+ also includes the usual equality axioms; clearly, these can be stated in positive sequent form as follows:

$$\Rightarrow\; t = t \qquad s = t \;\Rightarrow\; t = s \qquad s = t,\, t = r \;\Rightarrow\; s = r,$$
$$s_1 = t_1,\, s_2 = t_2 \;\Rightarrow\; s_1 s_2 = t_1 t_2 \qquad s = t,\, \mathsf{W}(s) \;\Rightarrow\; \mathsf{W}(t).$$

Let us now turn to the reformulation of the schemas ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\mathsf{W}}$) and ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\ell}$) of $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ notation induction on W and lexicographic induction on W, respectively. These will be replaced by suitable rules of inference in the Gentzen-style formulation of T^+. Let $A(u)$ be of the form $(\exists y \leq tu)B(u, y)$ for B being a positive and W free formula. Then an instance of the ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\mathsf{W}}$) notation induction rule is given as follows:

$$\frac{\Gamma,\, \mathsf{W}(u) \;\Rightarrow\; \mathsf{W}(tu),\, \Delta \quad \Gamma \;\Rightarrow\; A(\epsilon),\, \Delta \quad \Gamma,\, \mathsf{W}(u),\, A(u) \;\Rightarrow\; A(\mathsf{s}_i u),\, \Delta}{\Gamma,\, \mathsf{W}(s) \;\Rightarrow\; A(s),\, \Delta}$$

Here u denotes a fresh variable not occurring in Γ, Δ and i ranges over $0, 1$, i.e., the rule has four premises. Clearly, the main formulas of this rule are positive. We do not need to spell out the corresponding rule for ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\ell}$) lexicographic induction, since it is formulated in the very same manner except that it uses the successor s_{ℓ} instead of s_0 and s_1, thus only having three premises.

This ends the Gentzen-style reformulation of the non-logical axioms and rules of T^+. The logical axioms and rules of T^+ are just the usual ones for Gentzen's LK, cf. e.g. [35]. I.e., we have identity axioms, the well-known logical rules for introducing $\wedge, \vee, \neg, \rightarrow, \forall$ and $\exists$ on the right-hand side and on the left-hand side, the structural rules for weakening, exchange, and contraction, as well as the cut rule. In contrast to [35], however, we are using the so-called context-sharing or additive versions of these rules: this means that rules of inference with several premises are using the same context; we have already used this convention in the formulation of the induction rules above. To give a further example, the cut rule in its context-sharing version takes the form

$$\frac{\Gamma,\, A \;\Rightarrow\; \Delta \qquad \Gamma \;\Rightarrow\; A,\, \Delta}{\Gamma \;\Rightarrow\; \Delta}$$

As usual, we call the formula A the *cut formula* of this cut. We do not spell out all the rules of LK at the moment and refer the reader to the proofs of the realizability theorems in the following subsection, where some of these rules will be treated in all detail.

It should be clear that we have provided an adequate sequent-style reformulation of T^+; in particular, the axioms schemas ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\mathsf{W}}$) and ($\Sigma^{\mathsf{b}}_{\mathsf{W}}$-$\mathsf{I}_{\ell}$) as given in Section 3.2 of this paper are readily derivable by means of the corresponding rules of inference stated above, where as usual the presence of side formulas is crucial. In the following we often identify T^+ with its Gentzen-style version and write $\mathsf{T}^+ \vdash \Gamma \Rightarrow \Delta$ in order to express that the sequent $\Gamma \Rightarrow \Delta$ is derivable in T^+. Moreover, we will use the notation $\mathsf{T}^+ \vdash_{\star} \Gamma \Rightarrow \Delta$ if the sequent $\Gamma \Rightarrow \Delta$ has a proof in T^+ so that all cut formulas appearing in this proof are *positive*.

Due to the fact that all the main formulas of non-logical axioms and rules of T^+ are positive, we now obtain the desired partial cut elimination theorem for T^+. Its proof is immediate from the well-known proof of the cut elimination theorem for LK and is therefore omitted.

Theorem 12 (Partial cut elimination for T^+) *We have for all sequents $\Gamma \Rightarrow \Delta$ that $\mathsf{T}^+ \vdash \Gamma \Rightarrow \Delta$ entails $\mathsf{T}^+ \vdash_{\star} \Gamma \Rightarrow \Delta$.*

The following corollary directly follows from the above theorem and a quick inspection of the axioms and rules of T^+. It will be crucial for our realizability arguments below.

Corollary 13 *Assume that $\Gamma \Rightarrow \Delta$ is a sequent of positive formulas so that $\mathsf{T}^+ \vdash \Gamma \Rightarrow \Delta$. Then $\Gamma \Rightarrow \Delta$ has a T^+ derivation containing positive formulas only.*

6.2 The realizability theorems

In this subsection we use a realizability interpretation in the term model $\mathcal{M}(\lambda\eta)$ in order to determine the computational content of sequent-style derivations in the positive fragment of PT, PTLS, PS, and LS, respectively. We will show that the crucial realizing functions for our four systems belong to the corresponding function complexity classes on binary words, FPtime, FPtimeLinspace, FPspace, and FLinspace. As immediate corollaries of the four realizability theorems below we obtain the desired upper bounds for the provably total functions of PT, PTLS, PS, and LS.

The notion of realizability as well as the style and spirit of our realizability theorems are related to the work of Leivant [49], Schlüter [58], and Cantini

[16, 18], all three in the context of FPTIME. However, in contrast to these papers, we work in a bounded unramified setting. Moreover, and this is similar to [18, 58], we are able to realize directly quasi cut-free positive derivations in the *classical* sequent calculus. Finally, in order to find our realizing functions, we can make direct use of the function algebra characterizations of FPTIME, FPTIMELINSPACE, FPSPACE, and FLINSPACE given in Theorem 1; hence, direct reference to a machine model is not needed.

In fact, the above mentioned literature on realizability in an applicative context, especially in the classical setting, is clearly related to and inspired by older work on *witnessing* that has been used in classical fragments of arithmetic. In particular, Buss' witnessing technique (cf. Buss [8, 10, 11]) has been employed with great success in a variety of contexts. Another kind of witnessing is due to Sieg in his important work on "Herbrand analyses", cf. Sieg [60, 61], and also Buchholz and Sieg [7].

In our definition of realizability below we will make use of the open term model $\mathcal{M}(\lambda\eta)$ of T^+. This model is based on the usual $\lambda\eta$ reduction of the untyped lambda calculus (cf. [1, 38]) and exploits the well-known equivalence between combinatory logic with extensionality and $\lambda\eta$. In order to deal with the constants different from k and s, one extends $\lambda\eta$ reduction by the obvious reduction clauses for these new constants and checks that the so-obtained new reduction relation enjoys the Church Rosser property.[2]

The universe of the model $\mathcal{M}(\lambda\eta)$ now consists of the set of all $\mathcal{L}$ terms. Equality $=$ means reduction to a common reduct and W is interpreted as the set of all $\mathcal{L}$ terms t so that t reduces to a "standard" word $\overline{w}$ for some $w \in \mathbb{W}$. Finally, the constants are interpreted as indicated above and application of t to s is simply the term ts. As usual, we write $\mathcal{M}(\lambda\eta) \models A$ in order to express that the formula A is true in $\mathcal{M}(\lambda\eta)$.

We are now ready to turn to realizability. Our realizers $\rho, \sigma, \tau, \ldots$ are simply elements of the set $\mathbb{W}$ of binary words. We presuppose a low-level pairing operation $\langle\cdot,\cdot\rangle$ on $\mathbb{W}$ with associated projections $(\cdot)_0$ and $(\cdot)_1$; for definiteness, we assume that $\langle\cdot,\cdot\rangle$, $(\cdot)_0$, and $(\cdot)_1$ are in FPTIMELINSPACE. Further, for each natural number i let us write i_2 for the binary notation of i.

Since we are only interested in realizing *positive* derivations, we need to define realizability only for positive formulas. Accordingly, the crucial notion $\rho \ \mathbf{r} \ A$ ("ρ realizes A") for $\rho \in \mathbb{W}$ and A a positive formula, is given inductively in

[2]Actually, suitable interpretations for the constants $\mathsf{s}_\ell, \mathsf{p}_\ell, \mathsf{c}_\subseteq, \mathsf{l}_\mathsf{W}, *$ and $\times$ can also be given using the other constants of $\mathcal{L}$.

the following manner.

$$
\begin{array}{lll}
\rho \ \mathbf{r} \ \mathsf{W}(t) & \text{if} & \mathcal{M}(\lambda\eta) \models t = \overline{\rho}, \\
\rho \ \mathbf{r} \ (t_1 = t_2) & \text{if} & \rho = \epsilon \text{ and } \mathcal{M}(\lambda\eta) \models t_1 = t_2, \\
\rho \ \mathbf{r} \ (A \wedge B) & \text{if} & \rho = \langle \rho_0, \rho_1 \rangle \text{ and } \rho_0 \ \mathbf{r} \ A \text{ and } \rho_1 \ \mathbf{r} \ B, \\
\rho \ \mathbf{r} \ (A \vee B) & \text{if} & \rho = \langle i, \rho_0 \rangle \text{ and either } i = 0 \text{ and } \rho_0 \ \mathbf{r} \ A \text{ or} \\
 & & \qquad\qquad\qquad i = 1 \text{ and } \rho_0 \ \mathbf{r} \ B, \\
\rho \ \mathbf{r} \ (\forall x) A(x) & \text{if} & \rho \ \mathbf{r} \ A(u) \text{ for a fresh variable } u, \\
\rho \ \mathbf{r} \ (\exists x) A(x) & \text{if} & \rho \ \mathbf{r} \ A(t) \text{ for some term } t.
\end{array}
$$

If Δ denotes the sequence $A_1, \dots, A_n$ of positive formulas, then we say that ρ realizes the sequence Δ, in symbols, $\rho \ \mathbf{r} \ \Delta$, if $\rho = \langle i_2, \rho_0 \rangle$ for some $1 \leq i \leq n$ and $\rho_0 \ \mathbf{r} \ A_i$. Hence, according to the notion $\rho \ \mathbf{r} \ \Delta$, the sequence Δ is understood disjunctively, i.e. as the succedent of a given sequent.

It is important to note that in our definition of realizability, the realizers ρ mainly control information concerning the predicate W and, in addition, the usual information concerning conjunction and disjunction. However, the above notion of realizability *trivializes* quantifiers over arbitrary individuals.

The following properties concerning substitution will be crucial in the proof of the realizability theorem below. The proof of the following lemma is immediate from the definition of realizability and will therefore be omitted.

Lemma 14 (Substitution) *We have for all positive formulas A, all variables u and all terms s and t:*

1. *If $\rho \ \mathbf{r} \ A(t)$ and $\mathcal{M}(\lambda\eta) \models t = s$, then $\rho \ \mathbf{r} \ A(s)$.*
2. *If $\rho \ \mathbf{r} \ A(u)$, then $\rho \ \mathbf{r} \ A(t)$.*

Let us introduce some final piece of notation before we state the realizability theorem for PT. For an $\mathcal{L}$ formula A we write $A[\vec{u}]$ in order to express that all the free variables occurring in A are contained in the list $\vec{u}$. The analogous convention is used for finite sequences of $\mathcal{L}$ formulas.

Theorem 15 (Realizability for PT^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = A_1, \dots, A_n$ and assume that $\mathsf{PT}^+ \underset{\star}{\vdash} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in* FPTIME *so that we have for all terms $\vec{s}$ and all $\rho_1, \dots, \rho_n \in \mathbb{W}$:*

$$\textit{For all } 1 \leq i \leq n : \rho_i \ \mathbf{r} \ A_i[\vec{s}] \quad \Longrightarrow \quad F(\rho_1, \dots, \rho_n) \ \mathbf{r} \ \Delta[\vec{s}].$$

Proof We will prove our claim by induction on the length of quasi cut-free derivations of sequents of positive formulas in PT^+. In order to show that our realizing functions are in FPTIME we make use of the function algebra characterization of FPTIME given in Theorem 1. It is important that our realizing functions are invariant under substitutions of terms $\vec{s}$ for the free variables $\vec{u}$ in the sequent $\Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. This fact is always immediate and, therefore, in order to simplify notation, we often suppress substitutions in our discussion of the various axioms and rules below.

We start with a discussion of the logical axioms and rules of our sequent calculus LK. In the case of an identity axiom $A \Rightarrow A$ for A being a positive formula, we simply choose the function F with $F(\rho) = \langle 1, \rho \rangle$ as our realizing function so that our claim is immediate.

Let us turn to rules for conjunction introduction on the right and on the left. If our last inference is of the form

$$\frac{\Gamma \Rightarrow A, \Delta \qquad \Gamma \Rightarrow B, \Delta}{\Gamma \Rightarrow A \wedge B, \Delta},$$

and F_0 and F_1 are the two realizing functions for the left and the right premise of this rule, respectively, given to us by the induction hypothesis, then we define the realizing function F for the conclusion of the rule by

$$F(\vec{\rho}) \;=\; \begin{cases} F_0(\vec{\rho}) & \text{if } F_0(\vec{\rho})_0 \neq 1, \\ F_1(\vec{\rho}) & \text{if } F_0(\vec{\rho})_0 = 1 \text{ and } F_1(\vec{\rho})_0 \neq 1, \\ \langle 1, \langle F_0(\vec{\rho})_1, F_1(\vec{\rho})_1 \rangle \rangle & \text{otherwise.} \end{cases}$$

In the case of introduction of $\wedge$ on the left, i.e., if we have derived the sequent $\Gamma,\, A \wedge B \Rightarrow \Delta$ from $\Gamma, A \Rightarrow \Delta$ or $\Gamma,\, B \Rightarrow \Delta$, we choose $F(\vec{\rho}, \sigma)$ to be $F_0(\vec{\rho}, (\sigma)_0)$, respectively $F_0(\vec{\rho}, (\sigma)_1)$, for F_0 being the realizing function for the corresponding premise.

Next we discuss the rules for introducing a disjunction on the left and on the right. We first assume that our last inference is of the form

$$\frac{\Gamma \Rightarrow A, \Delta}{\Gamma \Rightarrow A \vee B, \Delta},$$

and we let F_0 denote the function given by the induction hypothesis. Then the realizing function F for the conclusion of this rule is given as follows:

$$F(\vec{\rho}) \;=\; \begin{cases} F_0(\vec{\rho}) & \text{if } F_0(\vec{\rho})_0 \neq 1, \\ \langle 1, \langle 0, F_0(\vec{\rho})_1 \rangle \rangle & \text{otherwise.} \end{cases}$$

The dual rule for introducing $\vee$ on the right is treated similarly. Now assume that our derivation ends with the rule

$$\frac{\Gamma,\, A \Rightarrow \Delta \qquad \Gamma,\, B \Rightarrow \Delta}{\Gamma,\, A \vee B \Rightarrow \Delta},$$

and let F_0 and F_1 be our realizing functions for the premises of this rule. Then we can simply define F by

$$F(\vec{\rho}, \sigma) \;=\; \begin{cases} F_0(\vec{\rho}, (\sigma)_1) & \text{if } (\sigma)_0 = 0, \\ F_1(\vec{\rho}, (\sigma)_1) & \text{otherwise.} \end{cases}$$

This ends our discussion of $\vee$ introduction. Observe that we do not have to consider introduction rules for negation and implication, since we are working in the positive fragment of PT^+.

We now address the quantification rules of LK. The introduction rules for universal quantification on the right and on the left have their usual form,

$$\frac{\Gamma \Rightarrow A(u),\, \Delta}{\Gamma \Rightarrow (\forall x)A(x),\, \Delta} \qquad \frac{\Gamma,\, A(t) \Rightarrow \Delta}{\Gamma,\, (\forall x)A(x) \Rightarrow \Delta},$$

for u a "fresh" variable and t an arbitrary term. Letting F_0 denote the function realizing the premise of these rules, it is straightforward to see that we can simply take $F = F_0$ for the function realizing the conclusion of the corresponding rule, since our definition of realizability trivializes quantifiers. In the case of the second of the above rules we further use the fact that our notion of realizability is closed under substitution (Lemma 14). Finally, it is easily seen that the choice $F = F_0$ also works equally well for the two introduction rules for the existential quantifiers, namely

$$\frac{\Gamma \Rightarrow A(t),\, \Delta}{\Gamma \Rightarrow (\exists x)A(x),\, \Delta} \qquad \frac{\Gamma,\, A(u) \Rightarrow \Delta}{\Gamma,\, (\exists x)A(x) \Rightarrow \Delta}.$$

In a further step we have to convince ourselves how to realize the structural rules of LK, namely *weakening, exchange* and *contraction.* As these rules are realized in a rather straightforward manner, we leave the details as an exercise to the devoted reader.

We conclude our discussion of the logical axioms and rules by considering the cut rule. Hence, by assumption, there exists a *positive* formula A so that our derivation ends by an application of the rule

$$\frac{\Gamma,\, A \Rightarrow \Delta \qquad \Gamma \Rightarrow A,\, \Delta}{\Gamma \Rightarrow \Delta}$$

By induction hypothesis we are given realizing functions F_0 and F_1 for the left and the right premise of this rule, respectively. We now obtain a realizing function F for $\Gamma \Rightarrow \Delta$ by setting

$$F(\vec{\rho}) = \begin{cases} F_1(\vec{\rho}) & \text{if } F_1(\vec{\rho})_0 \neq 1, \\ F_0(\vec{\rho}, F_1(\vec{\rho})_1) & \text{otherwise.} \end{cases}$$

Let us now turn to the non-logical axioms and rules of PT^+. First of all, it is quite easy to find realizing functions for the positive sequents corresponding to the axioms of $\mathsf{B}(*, \times)$. Instead of discussing all cases in detail we confine ourselves to looking at a few examples.

Clearly, sequents corresponding to true equations in the term model $\mathcal{M}(\lambda\eta)$ such as $\Rightarrow \mathsf{s}t_1t_2t_3 = t_1t_3(t_2t_3)$ are simply realized by the 0-ary function $F = \langle 1, \epsilon\rangle$. Further, for the two sequents given in the previous subsection for definition by cases on W we can simply take the two realizing functions $F_0(\rho, \sigma, \tau) = \langle 1, \epsilon\rangle$ as well as

$$F_1(\rho, \sigma) = \begin{cases} \langle 1, \epsilon\rangle & \text{if } \rho = \sigma, \\ \langle 2_2, \epsilon\rangle & \text{otherwise,} \end{cases}$$

respectively. Further, in order to be realize the two sequents corresponding to axioms (25) and (28) concerning the totality of word concatenation and word multiplication, namely

$$\mathsf{W}(s), \mathsf{W}(t) \Rightarrow \mathsf{W}(s{*}t), \qquad \mathsf{W}(s), \mathsf{W}(t) \Rightarrow \mathsf{W}(s{\times}t),$$

the two functions $F_0(\rho, \sigma) = \langle 1, \rho{*}\sigma\rangle$ and $F_1(\rho, \sigma) = \langle 1, \rho{\times}\sigma\rangle$ do the job. Also, it is easy to see how to realize the equality axioms. E.g., the sequent $s = t, \mathsf{W}(s) \Rightarrow \mathsf{W}(t)$ can be realized by the function $F(\rho, \sigma) = \sigma$.

Recall that PT^+ also includes an extensionality axiom for our notion of equality $=$, which we have formalized by the sequent $(\forall x)(sx = tx) \Rightarrow s = t$. Also this sequent is easily seen to be realizable by the function $F(\rho) = \langle 1, \epsilon\rangle$.

Let us now turn to the crucial part of the proof, namely the treatment of the rule for $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ notation induction on W. According to the four premises of $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ induction, we have quasi cut-free PT^+ derivations of the four sequents

$$\begin{aligned} &\Gamma, \mathsf{W}(u) \Rightarrow \mathsf{W}(tu), \Delta, \\ &\Gamma \Rightarrow A(\epsilon), \Delta, \\ &\Gamma, \mathsf{W}(u), A(u) \Rightarrow A(\mathsf{s}_i u), \Delta, \quad (i = 0, 1) \end{aligned}$$

for $A(u)$ being of the form $(\exists y \leq tu)B(u, y)$ with B positive and W free. Hence, the induction hypothesis guarantees the existence of four FPtime functions F, G_ϵ, G_0, and G_1 on $\mathbb{W}$, so that we have for all $\mathcal{L}$ terms $\vec{s}$ and all binary words $\vec{\rho}, \sigma, \tau$,

$$\begin{array}{ll}
(1) & \vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}] \quad \Longrightarrow \quad F(\vec{\rho},\sigma)\ \mathbf{r}\ \mathsf{W}(t[\vec{s}](\sigma)),\ \Delta[\vec{s}],^{3} \\
(2) & \vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}] \quad \Longrightarrow \quad G_\epsilon(\vec{\rho})\ \mathbf{r}\ A[\vec{s},\epsilon],\ \Delta[\vec{s}], \\
(3) & \vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}],\ \tau\ \mathbf{r}\ A[\vec{s},\sigma] \quad \Longrightarrow \quad G_i(\vec{\rho},\sigma,\tau)\ \mathbf{r}\ A[\vec{s},\mathsf{s}_i\sigma],\ \Delta[\vec{s}] \quad (i=0,1)
\end{array}$$

It is our aim to find a realizing function for the conclusion of the induction rule, i.e., a FPTIME function H so that we have for all $\vec{\rho}, \sigma$ in $\mathbb{W}$,

$$(4) \qquad \vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}] \quad \Longrightarrow \quad H(\vec{\rho},\sigma)\ \mathbf{r}\ A[\vec{s},\sigma],\ \Delta[\vec{s}].$$

Our desired word function H is defined for all $\vec{\rho}$ and σ in $\mathbb{W}$ as follows:

$$\begin{array}{rcl}
H(\vec{\rho},\epsilon) & = & G_\epsilon(\vec{\rho}), \\
H(\vec{\rho},\mathsf{s}_i\sigma) & = & \begin{cases} H(\vec{\rho},\sigma) & \text{if } H(\vec{\rho},\sigma)_0 \neq 1, \\ F(\vec{\rho},\sigma) & \text{if } H(\vec{\rho},\sigma)_0 = 1 \text{ and } F(\vec{\rho},\sigma)_0 \neq 1, \\ G_i(\vec{\rho},\sigma,H(\vec{\rho},\sigma)_1) & \text{otherwise.} \end{cases}
\end{array}$$

It is now a matter of routine to check (4) by (meta) notation induction on σ, using our assertions (1)–(3) from the induction hypothesis.

It still remains to check that the function H is indeed in FPTIME. Clearly, H is defined by recursion on notation from functions which are already known to be in FPTIME and, hence, it is sufficient to provide a suitable bound for H; of course it is enough to bound $H(\vec{\rho},\sigma)$ under the assumption that $\vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}]$. Looking at our recursive definition of H, it is clear that H stays constant whenever we enter the first or the second case of our three-fold case distinction, so that bounding will be immediate from our discussion below. Further, when setting

$$(5) \qquad H(\vec{\rho},\mathsf{s}_i\sigma) = G_i(\vec{\rho},\sigma,H(\vec{\rho},\sigma)_1)$$

in the third case, we know that $H(\vec{\rho},\sigma)_0 = 1$ *and* $F(\vec{\rho},\sigma)_0 = 1$. Using (4) and (1) together with our assumption $\vec{\rho}\ \mathbf{r}\ \Gamma[\vec{s}]$ this means in particular that

$$(6) \qquad H(\vec{\rho},\sigma)_1\ \mathbf{r}\ A[\vec{s},\sigma] \quad \text{and} \quad F(\vec{\rho},\sigma)_1\ \mathbf{r}\ \mathsf{W}(t[\vec{s}](\sigma)).$$

But now we have to recall that the formula $A[\vec{s},\sigma]$ has the shape

$$(\exists y \in \mathsf{W})[y \leq t[\vec{s}](\sigma) \ \wedge\ B[\vec{s},y,\sigma]],$$

[3]Temporarily in this proof, if $\Gamma = C_1,\dots,C_m$ is a sequence of formulas contained in the antecedent of a sequent, then we write $\rho_1,\dots,\rho_m\ \mathbf{r}\ \Gamma$ if for all $1 \leq i \leq m$, $\rho_i\ \mathbf{r}\ C_i$.

with B positive and W free; hence, the only occurrence of W in $A[\vec{s}, \sigma]$ stems from the leading bounded existential quantifier. But the bounding term $t[\vec{s}](\sigma)$ of this quantifier evaluates to $F(\vec{\rho}, \sigma)_1$ in $\mathcal{M}(\lambda\eta)$ according to (6). It is now easy to see that $H(\vec{\rho}, \sigma)_1$ is bounded by a *linear function* L in the length of $F(\vec{\rho}, \sigma)_1$; this only uses some obvious properties of our low level pairing function. It follows from these considerations that if we define $H(\vec{\rho}, \mathsf{s}_i\sigma)$ by (5) according to the third case in our case distinction, then it is clearly bounded. This ends our considerations concerning the bounding of the function H.

We have shown that the conclusion of the $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ notation induction rule can be realized by a FPTIME function H. This ends our discussion of the induction rule and, in fact, also the proof of the realizability theorem for PT^+. $\square$

The following corollary is immediate from our realizability theorem for PT^+ as well as the partial cut elimination theorem for PT^+ (Theorem 12). It shows that the provably total functions of PT^+ are contained in FPTIME.

Corollary 16 *Let t be a closed $\mathcal{L}$ term and assume that*

$$\mathsf{PT}^+ \vdash \mathsf{W}(u_1), \dots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \dots u_n),$$

for distinct variables $u_1, \dots, u_n$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in FPTIME *so that we have for all words $w_1, \dots, w_n$ in $\mathbb{W}$,*

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \dots \overline{w}_n = \overline{F(w_1, \dots, w_n)}.$$

Proof Assuming that we have a closed $\mathcal{L}$ term t so that the sequent

$$\mathsf{W}(u_1), \dots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \dots u_n)$$

is provable in PT^+, we know that by partial cut elimination, this sequent has a proof using positive cut formulas only. Hence, our theorem provides a function G in FPTIME so that we have for all $\mathcal{L}$ terms $s_1, \dots, s_n$ and all words $\rho_1, \dots, \rho_n$ in $\mathbb{W}$,

$$G(\rho_1, \dots, \rho_n)_1 \ \mathbf{r} \ \mathsf{W}(ts_1 \dots s_n),$$

whenever $\rho_i \, \mathbf{r} \, \mathsf{W}(s_i)$ for all $1 \leq i \leq n$. If we now set for given words $w_1, \dots, w_n$ in $\mathbb{W}$,

$$s_i = \overline{w}_i, \ \rho_i = w_i, \ \text{ and } F(w_1, \dots, w_n) = G(w_1, \dots, w_n)_1,$$

then the assertion of our corollary is immediate. $\square$

This ends our discussion of the realizability theorem for PT^+ and its crucial consequences. Turning to the realizability theorem for PTLS^+, note that the only difference between PT^+ and $\mathsf{PTLS}^+$is the presence of word multiplication $\times$ in PT^+. Hence, the proof of the following theorem is literally the same as the proof of the realizability theorem for PT^+, but since $\times$ does not need to be realized, the corresponding realizing function is indeed in FPtimeLinspace, according to the function algebra characterization of FPtimeLinspace given in Theorem 1. Again we can derive the desired corollary about the provably total functions of PTLS^+.

Theorem 17 (Realizability for PTLS^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = A_1, \ldots, A_n$ and assume that $\mathsf{PTLS}^+ \vdash_{\star} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in* FPtimeLinspace *so that we have for all terms $\vec{s}$ and all $\rho_1, \ldots, \rho_n \in \mathbb{W}$:*

$$\text{For all } 1 \leq i \leq n : \rho_i \ \mathbf{r}\ A_i[\vec{s}] \quad \Longrightarrow \quad F(\rho_1, \ldots, \rho_n) \ \mathbf{r}\ \Delta[\vec{s}].$$

Corollary 18 *Let t be a closed $\mathcal{L}$ term and assume that*

$$\mathsf{PTLS}^+ \vdash \mathsf{W}(u_1), \ldots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \ldots u_n),$$

for distinct variables $u_1, \ldots, u_n$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in FPtimeLinspace *so that we have for all words $w_1, \ldots, w_n$ in $\mathbb{W}$,*

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \ldots \overline{w}_n = \overline{F(w_1, \ldots, w_n)}.$$

Let us now discuss the realizability theorems for the two systems PS^+ and LS^+. Indeed, also the proof of these theorems runs very analogous to the proof of the realizability theorem for PT^+. The crucial difference between PS^+ and PT^+ lies in the fact that PS^+ contains lexicographic induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$, instead of the schema $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ of notation induction on W present in PT^+. The only difference in the realization of the corresponding rules of inference in the sequent-style setting is that one requires bounded lexicographic recursion (BRL) in order to realize the $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$ rule, where, as we have seen above, bounded recursion on notation (BRN) was needed for the realization of the $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ induction rule. Otherwise, the proof of the realizability theorem for PS^+ is identical to the one for PT^+. Hence, using the characterization of FPspace stated in Theorem 1, we are thus in a position to spell out the following theorem together with its expected corollary.

Theorem 19 (Realizability for PS^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = A_1, \ldots, A_n$ and assume that $\mathsf{PS}^+ \vdash_{\star} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then* *there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in* FPspace *so that we have for all terms $\vec{s}$ and all $\rho_1, \dots, \rho_n \in \mathbb{W}$:*

$$\text{For all } 1 \leq i \leq n : \rho_i \ \mathbf{r} \ A_i[\vec{s}] \quad \Longrightarrow \quad F(\rho_1, \dots, \rho_n) \ \mathbf{r} \ \Delta[\vec{s}].$$

Corollary 20 *Let t be a closed $\mathcal{L}$ term and assume that*

$$\mathsf{PS}^+ \vdash \mathsf{W}(u_1), \dots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \dots u_n),$$

for distinct variables $u_1, \dots, u_n$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in FPspace *so that we have for all words $w_1, \dots, w_n$ in $\mathbb{W}$,*

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \dots \overline{w}_n = \overline{F(w_1, \dots, w_n)}.$$

As above, if word multiplication $\times$ is absent, then the proof for PS^+ actually produces realizing functions in FLinspace. Thus we obtain the following realizability theorem for the system LS^+.

Theorem 21 (Realizability for LS^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = A_1, \dots, A_n$ and assume that $\mathsf{LS}^+ \underset{\star}{\vdash} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in* FLinspace *so that we have for all terms $\vec{s}$ and all $\rho_1, \dots, \rho_n \in \mathbb{W}$:*

$$\text{For all } 1 \leq i \leq n : \rho_i \ \mathbf{r} \ A_i[\vec{s}] \quad \Longrightarrow \quad F(\rho_1, \dots, \rho_n) \ \mathbf{r} \ \Delta[\vec{s}].$$

Corollary 22 *Let t be a closed $\mathcal{L}$ term and assume that*

$$\mathsf{LS}^+ \vdash \mathsf{W}(u_1), \dots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \dots u_n),$$

for distinct variables $u_1, \dots, u_n$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in FLinspace *so that we have for all words $w_1, \dots, w_n$ in $\mathbb{W}$,*

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \dots \overline{w}_n = \overline{F(w_1, \dots, w_n)}.$$

The results of this section can now be combined with our lower bound results given in Theorem 9; hence, we have established the following crucial theorem concerning the provably total functions of PT, PTLS, LS, and PS.

Theorem 23 *We have the following proof-theoretic results:*

1. *The provably total functions of* PT *coincide with* FPtime.
2. *The provably total functions of* PTLS *coincide with* FPtimeLinspace.
3. *The provably total functions of* PS *coincide with* FPspace.
4. *The provably total functions of* LS *coincide with* FLinspace.

Moreover, this theorem holds true in the presence of totality of application (Tot) *and extensionality of operations* (Ext).

7 Further applicative systems

It is the aim of this section to consider further natural applicative systems for various classes of computable functions. We start with the system PH which is closely related to the polynomial time hierarchy Ph. The second subsection is concerned with applicative systems for the primitive recursive functions and, finally, in the last subsection we make some remarks concerning an applicative setting which is of the same strength as Peano arithmetic PA. We will see that the techniques developed in this paper so far extend in a straightforward manner to the various systems considered in this section.

7.1 A type two functional for bounded quantification

In this subsection we consider a natural type two functional π which allows for the elimination of bounded quantifiers. Using the techniques of the previous section we will show that the provably total functions of the theory PT augmented by π are exactly the functions on $\mathbb{W}$ in the function polynomial time hierarchy FPh.

It is worth mentioning at this point that the formulation and spirit of the π functional is similar to the non-constructive μ operator which has been studied extensively in the applicative context, cf. the papers Feferman and Jäger [30, 31], Glass and Strahm [36], Jäger and Strahm [45], Marzetta and Strahm [52], and Strahm [63]. In contrast to π, the operator μ tests for *unbounded* quantification and, hence, is much stronger than the π functional. The applicative axiomatization of the two functionals, however, is completely analogous.

As usual, a function F on the binary words $\mathbb{W}$ is defined to be in the *(function) polynomial time hierarchy* FPh if F is computable in polynomial time using finitely many oracles from the Meyer-Stockmeyer polynomial time hierarchy Ph on $\mathbb{W}$. It is well-known how to extend Cobham's function algebra characterization of FPtime so as to capture FPh: one simply closes the Cobham algebra under bounded quantification. In the sequel we let (BQ) denote the operator which maps an $(n{+}1)$-ary function F on $\mathbb{W}$ to the $(n{+}1)$-ary function $\mathsf{BQ}(F)$, which is given for all $\vec{x}, y \in \mathbb{W}$ as follows:

$$\mathsf{BQ}(F)(\vec{x}, y) \ := \begin{cases} 0 & \text{if } (\exists z \le y)(F(\vec{x}, z) = 0), \\ 1 & \text{otherwise.} \end{cases}$$

The following theorem is folklore, cf. Clote's survey article [19] on function algebras and computation models.

Theorem 24 *We have the following function algebra characterization:*

$$[\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1, *, \times; \mathsf{COMP}, \mathsf{BRN}, \mathsf{BQ}] = \mathrm{FP}\textsc{h}.$$

For the formulation of our type two functional for bounded quantification in the applicative setting, we assume that the applicative language $\mathcal{L}$ is extended by a new constant π. The axioms for π are divided into (π.1) and (π.2): the first axiom claims that for a given total operation f on W and an $a \in \mathsf{W}$, it is always the case that $\pi f a$ is a word whose length is bounded by the length of a; the second axioms expresses, in addition, that $\pi f a$ is a zero of f provided that there exists a word $x \le a$ with $fx = 0$. Hence, given that $f : \mathsf{W} \to \mathsf{W}$ and $a \in \mathsf{W}$, we have that indeed $(\exists x \le a)(fx = 0)$ *is equivalent to* $f(\pi f a) = 0$, i.e., bounded quantifiers can be eliminated by means of π.

The type two functional π for bounded quantification

(π.1) $\quad f : \mathsf{W} \to \mathsf{W} \wedge a \in \mathsf{W} \;\to\; \pi f a \in \mathsf{W} \wedge \pi f a \le a$

(π.2) $\quad f : \mathsf{W} \to \mathsf{W} \wedge a \in \mathsf{W} \wedge (\exists x \le a)(fx = 0) \;\to\; f(\pi f a) = 0$

We now define the $\mathcal{L}$ theory PH to be simply PT plus the two axioms (π.1) and (π.2). We aim at showing that the provably total functions of PH are exactly the functions in the function polynomial time hierarchy FPh.

Clearly, we can make use of the function algebra characterization of FPh given in the theorem above in order to show that the provably total functions of PH contain FPh: with the help of π we have closure under bounded quantification and, moreover, due to Lemma 4 we know that in PH closure under bounded recursion on notation is available. Hence, we can state the following theorem.

Theorem 25 *The provably total functions of* PH *include* FPh.

Indeed, let us mention that it is possible to show that Ferreira's system Σ^b_∞-NIA (cf. Ferreira [34]) or, equivalently, Buss' system S_2 (cf. Buss [8]) are directly contained in PH.

In order to show that the lower bound stated in the above theorem is sharp, we can make use in a straightforward manner of the partial cut elimination and realizability techniques introduced in the previous section. In the following we sketch the main new steps of this procedure.

As above, we provide an upper bound directly for the system PH^+, i.e., the extension of PH by totality and extensionality. The Gentzen-style reformulation of PH^+ simply extends the Gentzen-style version of PT^+ by two new rules corresponding to the axioms (π.1) and (π.2) for π. As expected, in these rules u denotes a fresh variable.

$$\frac{\Gamma,\, \mathsf{W}(u) \;\Rightarrow\; \mathsf{W}(tu),\, \Delta}{\Gamma,\, \mathsf{W}(s) \;\Rightarrow\; \mathsf{W}(\pi ts) \wedge \pi ts \leq s,\, \Delta} \quad (\pi.1)$$

$$\frac{\Gamma,\, \mathsf{W}(u) \;\Rightarrow\; \mathsf{W}(tu),\, \Delta}{\Gamma,\, \mathsf{W}(s),\, (\exists x \leq s)(tx = 0) \;\Rightarrow\; t(\pi ts) = 0,\, \Delta} \quad (\pi.2)$$

We observe that the main formulas of both rules are *positive*, so that the partial cut elimination theorem for PT^+ (Theorem 12) readily extends to PH^+. Hence, we can assume that PH^+ derivations of sequents of positive formulas contain cuts with positive cut formulas only.

In the sequel we want to use the same notion of realizability as in the previous section. Hence, we have to extend our open term model $\mathcal{M}(\lambda\eta)$ so as to incorporate the new constant π. The informal interpretation of πfa is simply the least $x \leq a$ so that $fx = 0$, if such an x exists, and ϵ otherwise.[4] Formally in $\mathcal{M}(\lambda\eta)$, we can either write down appropriate reduction rules for π or use recursion in $\mathcal{M}(\lambda\eta)$ in order to define π directly. The realizability theorem for PH^+ is now spelled out in the expected manner.

Theorem 26 (Realizability for PH^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = A_1, \ldots, A_n$ and assume that $\mathsf{PH}^+ \mathrel{\vert\!\!\frac{}{\star}} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in* FPh *so that we have for all terms $\vec{s}$ and all $\rho_1, \ldots, \rho_n \in \mathbb{W}$:*

$$\textit{For all } 1 \leq i \leq n : \rho_i \;\mathbf{r}\; A_i[\vec{s}] \quad\Longrightarrow\quad F(\rho_1, \ldots, \rho_n) \;\mathbf{r}\; \Delta[\vec{s}].$$

Proof In addition to the proof of the realizability theorem for PT^+ we only have to show how to deal with the two rules (π.1) and (π.2). For that purpose let us assume that we have a quasi cut free derivation of the sequent

$$\Gamma,\, \mathsf{W}(u) \;\Rightarrow\; \mathsf{W}(tu),\, \Delta,$$

and let F_0 denote the function in FPh which is given to us by the induction hypothesis. In case of (π.1) it is not difficult to check that the following function F can be used as a realizing function for the conclusion of this rule.

$$F(\vec{\rho}, \sigma) \;=\; \begin{cases} \langle 1, \langle (\mu\tau \leq \sigma . F_0(\vec{\rho}, \tau)_1 = 0), \epsilon \rangle\rangle & \text{if } (\forall \tau \leq \sigma) F_0(\vec{\rho}, \tau)_0 = 1, \\ F_0(\vec{\rho}, (\mu\tau \leq \sigma) F_0(\vec{\rho}, \tau)_0 \neq 1) & \text{otherwise} \end{cases}$$

[4] Leastness is always understood in the sense of the lexicographic ordering of the full binary tree. In the sequel we use the notation $(\mu x \leq a)R(x)$ to denote the least $x \leq a$ satisfying $R(x)$ if it exists, and ϵ otherwise.

It is easy to see that F is in FPH, since the functions in the polynomial time hierarchy are clearly closed under bounded minimization. In the case of the rule for $(\pi.2)$ the realizing function F for its conclusion can be chosen as follows. Again it is easy to see that this F is in FPH.

$$F(\vec{\rho}, \sigma, \sigma') = \begin{cases} \langle 1, \epsilon \rangle & \text{if } (\forall \tau \leq \sigma) F_0(\vec{\rho}, \tau)_0 = 1, \\ F_0(\vec{\rho}, (\mu\tau \leq \sigma) F_0(\vec{\rho}, \tau)_0 \neq 1) & \text{otherwise} \end{cases}$$

This ends our short discussion of the proof of the realizability theorem for the system PH^+. □

As above, we can now derive the following crucial corollary.

Corollary 27 *Let t be a closed $\mathcal{L}$ term and assume that*

$$\mathsf{PH}^+ \vdash \mathsf{W}(u_1), \ldots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \ldots u_n),$$

for distinct variables $u_1, \ldots, u_n$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in FPH *so that we have for all words $w_1, \ldots, w_n$ in $\mathbb{W}$,*

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \ldots \overline{w}_n = \overline{F(w_1, \ldots, w_n)}.$$

This last corollary combined with Theorem 25 yields the following main result of this subsection.

Theorem 28 *The provably total functions of* PH *coincide with* FPH. *In addition, this theorem holds true in the presence of totality of application* (Tot) *and extensionality of operations* (Ext).

7.2 Positive induction equals primitive recursion

In this subsection we briefly examine the effect of replacing our bounded induction principles $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$ and $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$ by the schema of induction for arbitrary *positive* formulas. We will show that the corresponding applicative framework characterizes exactly the class of primitive recursive functions. This result is previously due to Cantini [15] This result is previously due to Cantini [15][5]. However, the proof given here is new and quite different from the techniques used by Cantini.

The primitive recursive functions FPRIM on $\mathbb{W}$ are generated from the usual initial functions by closing under composition and recursion on notation

[5]Actually, Cantini establishes a slightly stronger theorem in the sense that he also allows *negative equations* to occur in induction formulas.

(RN), where (RN) is simply (BRN) without the bounding condition. Hence, using our function algebra notation, FPRIM is defined to be the function algebra $[\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1; \mathsf{COMP}, \mathsf{RN}]$. Denoting by (RL) the corresponding schema of unbounded lexicographic recursion, it is well known that indeed

$$[\epsilon, \mathsf{I}, \mathsf{s}_\ell; \mathsf{COMP}, \mathsf{RL}] = [\epsilon, \mathsf{I}, \mathsf{s}_0, \mathsf{s}_1; \mathsf{COMP}, \mathsf{RN}].$$

Hence, it does not matter whether we use lexicographic or notation recursion in the context of unbounded recursion schemas.

Let us now turn to a natural applicative framework PR capturing FPRIM. The schema of positive notation induction on W, $(\mathsf{Pos\text{-}I_W})$, includes for each formula $A(x)$ in the class Pos,

$$(\mathsf{Pos\text{-}I_W}) \qquad A(\epsilon) \wedge (\forall x \in \mathsf{W})(A(x) \rightarrow A(\mathsf{s}_0 x) \wedge A(\mathsf{s}_1 x)) \ \rightarrow \ (\forall x \in \mathsf{W})A(x)$$

The schema of positive lexicographic induction on W, $(\mathsf{Pos\text{-}I}_\ell)$, is stated accordingly. The applicative theory PR is now defined to be the theory B plus positive notation induction on W, $(\mathsf{Pos\text{-}I_W})$. Observe that we do not include $*$ and $\times$ in PR since these are easily definable as we will see now.

As can be expected, it is possible represent recursion on notation in PR in a very direct and natural way, by referring to the recursion theorem of B and exploiting $(\mathsf{Pos\text{-}I_W})$. In particular, we obtain in a straightforward manner the following unbounded analogue of Lemma 4; its proof is an obvious adaptation of the proof of Lemma 4 and, therefore, is left to the reader.

Lemma 29 *There exists a closed $\mathcal{L}$ term $\tilde{\mathsf{r}}_\mathsf{W}$ so that* PR *proves*

$$f : \mathsf{W} \rightarrow \mathsf{W} \wedge g : \mathsf{W}^3 \rightarrow \mathsf{W} \ \rightarrow$$
$$\begin{cases} \tilde{\mathsf{r}}_\mathsf{W} fg : \mathsf{W}^2 \rightarrow \mathsf{W} \wedge \\ x \in \mathsf{W} \wedge y \in \mathsf{W} \wedge y \neq \epsilon \wedge h = \tilde{\mathsf{r}}_\mathsf{W} fg \ \rightarrow \\ \qquad hx\epsilon = fx \wedge hxy = gxy(hx(\mathsf{p_W} y)) \end{cases}$$

Corollary 30 *The provably total functions of* PR *include* FPRIM.

Indeed, PR does not only establish the convergence of each primitive recursive function, but it also interprets in a straightforward manner the subsystem of Peano arithmetic PA which is based on the schema of complete induction for Σ_1 formulas. The latter system is well-known to be a conservative extension with respect to Π_2 statements of primitive recursive arithmetic PRA as was shown by Parsons [54].

Before we turn to the upper bound of PR let us quickly address the question of whether it matters if we include $(\mathsf{Pos\text{-}I_W})$ or $(\mathsf{Pos\text{-}I}_\ell)$ in our definition of PR. As our discussion above concerning the corresponding function algebras suggests, there should be no difference, and indeed this is confirmed by the following folklore lemma, whose proof is left to the reader as an exercise.

Lemma 31 *We have that* $(\mathsf{Pos\text{-}I}_\ell)$ *and* $(\mathsf{Pos\text{-}I_W})$ *are equivalent over our base theory* B.

Clearly, this lemma shows that in the theory PR we have available the lexicographic analogue of Lemma 29.

The final part of this subsection is devoted to showing that the provably total functions of PR do not go beyond the primitive recursive functions FPrim on $\mathbb{W}$. Again our realizability techniques work in a perspicuous manner. We first reformulate the system PR^+, i.e., $\mathsf{PR} + (\mathsf{Tot}) + (\mathsf{Ext})$, in sequent style. Positive induction on notation $(\mathsf{Pos\text{-}I_W})$ is stated as a rule in the same way as for the system PT^+, but of course without the premise concerning the totality of a bounding function. Partial cut elimination for PR^+ works as before. As to the realizability theorem, its proof is literally the same as the proof of the realizability theorem for PT^+, with the only difference that in the treatment of the notation induction rule, we have no bounding information available and, hence, we can only conclude that the corresponding function is primitive recursive.

Theorem 32 (Realizability for PR^+) *Let* $\Gamma \Rightarrow \Delta$ *be a sequent of positive formulas with* $\Gamma = A_1, \ldots, A_n$ *and assume that* $\mathsf{PR}^+ \underset{\star}{\vdash} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. *Then there exists a function* $F : \mathbb{W}^n \to \mathbb{W}$ *in* FPrim *so that we have for all terms* $\vec{s}$ *and all* $\rho_1, \ldots, \rho_n \in \mathbb{W}$:

$$\textit{For all } 1 \le i \le n : \rho_i \mathbf{\ r\ } A_i[\vec{s}] \quad \Longrightarrow \quad F(\rho_1, \ldots, \rho_n) \mathbf{\ r\ } \Delta[\vec{s}].$$

Corollary 33 *Let* t *be a closed* $\mathcal{L}$ *term and assume that*

$$\mathsf{PR}^+ \vdash \mathsf{W}(u_1), \ldots, \mathsf{W}(u_n) \Rightarrow \mathsf{W}(tu_1 \ldots u_n),$$

for distinct variables $u_1, \ldots, u_n$. *Then there exists a function* $F : \mathbb{W}^n \to \mathbb{W}$ *in* FPrim *so that we have for all words* $w_1, \ldots, w_n$ *in* $\mathbb{W}$,

$$\mathcal{M}(\lambda\eta) \models t\overline{w}_1 \ldots \overline{w}_n = \overline{F(w_1, \ldots, w_n)}.$$

From this corollary and Corollary 30 we are now in a position to state the following crucial theorem concerning the provably total functions of PR. As we have noted above, this theorem has previously been obtained by Cantini [15], using a quite different argument.

Theorem 34 *The provably total functions of* PR *coincide with* $\mathrm{FP{\scriptsize RIM}}$. *In addition, this theorem holds true in the presence of totality of application* (Tot) *and extensionality of operations* (Ext).

7.3 Full induction and Peano arithmetic

A further natural strengthening of our applicative framework consists in allowing induction on W for arbitrary formulas in the language $\mathcal{L}$. Using known techniques, it easily follows that the so-obtained applicative systems have the same proof-theoretic strength as Peano arithmetic PA.

By ($\mathcal{L}$-$\mathsf{I_W}$) and ($\mathcal{L}$-I_ℓ) we denote the schema of notation induction and lexicographic induction on W, respectively, for arbitrary formulas of our applicative language $\mathcal{L}$. With the same argument as in Lemma 31 above one establishes that ($\mathcal{L}$-$\mathsf{I_W}$) and ($\mathcal{L}$-I_ℓ) are equivalent over the base theory B. For an interpretation of $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I_W})$ or $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I}_\ell)$ in Peano arithmetic PA, one makes use of an inner model construction, formalizing the standard recursion-theoretic model *PRO* of B, cf. e.g. Feferman and Jäger [30] for a similar argument. The so-obtained interpretation yields that the provably total functions of $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I_W})$ and $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I}_\ell)$ are exactly the α recursive functions for α less than PA's proof-theoretic ordinal ε_0.

The interpretation of $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I_W})$ or $\mathsf{B} + (\mathcal{L}\text{-}\mathsf{I}_\ell)$ can also be strengthened so as to include the axiom of totality (Tot) and the axiom of extensionality (Ext). In this case, one simply formalizes the standard term model $\mathcal{M}(\lambda\eta)$ of $\mathsf{B} + (\mathsf{Tot}) + (\mathsf{Ext})$ in PA, cf. Cantini [14] or Jäger and Strahm [45] for more details.

Let us conclude this section by noting that similar inner model constructions are of no use in order to establish upper bounds e.g. for the system PR: the reason is that in induction formulas in PR arbitrary unbounded universal quantifiers over individuals are allowed, which makes an embedding in, say, primitive recursive arithmetic PRA extended by Σ_1 induction impossible.

8 Concluding remarks

In this article we have presented a series of natural applicative systems of various bounded complexities. In particular, we have elucidated frameworks for the functions on binary words computable in polynomial time, polynomial time *and* linear space, polynomial space, linear space, as well as the polynomial time hierarchy. Our systems can be viewed as natural applicative analogues of various bounded arithmetics; this is witnessed by the fact that the latter are directly embeddable into various applicative settings. A further distinguished feature of applicative theories is that they allow for a very direct treatment of higher types issues: we have seen that even higher order systems such as Cook and Urquhart's PV^ω are directly contained in the applicative theory PT for the polynomial time computable functions.

Apart from the world of bounded recursion schemas, bounded arithmetic and bounded applicative theories there is the world of so-called *tiered systems* in the sense of Cook and Bellantoni (cf. e.g. [5]) and Leivant (cf. e.g. [49, 51]). Crucial for this approach to characterizing complexities is a strictly predicative regime which distinguishes between different uses of variables in induction and recursion schemas, thus severely restricting the definable or provably total functions in various unbounded formalisms. In our applicative setting such a "predicativization" amounts to distinguishing between (at least) two sorts or types of binary words W_0 and W_1, say, where induction over W_1 is allowed for formulas which are positive and do not contain W_1, cf. Cantini [16, 18] for such systems.

Unarguably, the tiered approach to complexity has led to numerous highly interesting and intrinsic recursion-theoretic and also proof-theoretic characterizations of complexity classes, which might lead to new subrecursive programming paradigms. Also, higher type issues have recently been a subject of interest in this area, cf. e.g. Leivant [50], Bellantoni, Niggl, Schwichtenberg [6], and Hofmann [40]. In spite of its elegance, it has to be mentioned that the tiered or ramified approach also has its drawbacks. First of all, there is the general observation that reasoning in a system with ramifications can be very difficult: for example, dealing with two tiers W_0 and W_1 only, one has to take into account four kinds of functions from binary words to binary words, which are not closed under composition, of course. Secondly, the strict predicative regime disallows the direct formulation of many natural algorithms, especially those obtained by various kinds of nested recursions, cf. Hofmann [39] for a discussion. And thirdly, it is not at all clear how modern tiered systems relate to the more traditional bounded subsystems of first and higher order arithmetic.

Taking up these points of criticism in the context of the bounded world, of course one has to pay a price in order to avoid ramifications and to deal only with one type W of binary words. Namely, the systems discussed in this article include initial functions such as word concatenation and word multiplication as well as recursions and inductions need to be bounded. On the other hand, nesting recursions is generally easy and in many cases it is also not difficult to provide the necessary bounding information. Hence, both the bounded and the tiered approach have their pros and cons. Summing up, in our opinion it is worth exploring both the bounded *and* the ramified world, and it would be especially interesting to find out more about the exact relationship between these two worlds.

Coming back to the work and results achieved in this article, let us briefly address some directions for future research. Certainly, there is the need to further study and elucidate the role of higher type functionals in the various settings that we have been considering in this paper. We have done a first step in this direction and shown that indeed *the provably total type two functionals of the classical theory* PT *coincide with the basic feasible functionals of type two*, and we conjecture that this results holds true at all higher types. The proof that a provably total type two functional of PT is basic feasible is based on a refinement of the realizability theorem for PT established in this paper. Details are given in a successor to this paper, Strahm [65].

Finally, a further important research project consists in considering extensions of the applicative systems of this article by adding suitable versions of flexible typing and naming in the spirit of explicit mathematics in order to answer the question of what type existence principles can live in a, say, feasible setting of explicit mathematics. We believe that the formalisms designed in this paper should help in finding suitable frameworks of "bounded explicit mathematics".

References

[1] Barendregt, H. P. *The Lambda Calculus*, revised ed. North Holland, Amsterdam, 1984.

[2] Beckmann, A. *Separating fragments of bounded arithmetic.* PhD thesis, Universität Münster, 1996.

[3] Beeson, M. J. *Foundations of Constructive Mathematics: Metamathematical Studies.* Springer, Berlin, 1985.

[4] Beeson, M. J. Proving programs and programming proofs. In *Logic, Methodology and Philosophy of Science VII*, Barcan Marcus et. al., Ed. North Holland, Amsterdam, 1986, pp. 51–82.

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

[6] Bellantoni, S., Niggl, K.-H., and Schwichtenberg, H. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic 104*, 1–3 (2000), 17–30.

[7] Buchholz, W., and Sieg, W. A note on polynomial time computable arithmetic. In *Logic and Computation, Proceedings of a Workshop held at Carnegie Mellon University, 1987*, W. Sieg, Ed., vol. 106 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, 1990, pp. 51–55.

[8] Buss, S. R. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986.

[9] Buss, S. R. Axiomatizations and conservation results for fragments of bounded arithmetic. In *Logic and Computation, Proceedings of a Workshop held at Carnegie Mellon University, 1987*, W. Sieg, Ed., vol. 106 of *Contemporary Mathematics*. American Mathematical Society, Providence, Rhode Island, 1990, pp. 57–84.

[10] Buss, S. R. The witness function method and fragments of Peano arithmetic. In *Proceedings of the Ninth International Congress on Logic, Methodology and Philosophy of Science, Uppsala, Sweden, August 7–14, 1991*, D. Prawitz, B. Skyrms, and D. Westerdåhl, Eds. Elsevier, North Holland, Amsterdam, 1994, pp. 29–68.

[11] Buss, S. R. First-order proof theory of arithmetic. In *Handbook of Proof Theory*, S. R. Buss, Ed. Elsevier, 1998, pp. 79–147.

[12] Cantini, A. Choice and uniformity in weak applicative theories. Invited talk, Logic Colloquium 2001, 31 pages. Submitted for publication.

[13] Cantini, A. On the computational content of theories of operations with total application, June 1995. Handwritten notes.

[14] Cantini, A. *Logical Frameworks for Truth and Abstraction*. North-Holland, Amsterdam, 1996.

[15] Cantini, A. Proof-theoretic aspects of self-referential truth. In *Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995*, Maria Luisa Dalla Chiara et. al., Ed., vol. 1. Kluwer, September 1997, pp. 7–27.

[16] Cantini, A. Characterizing poly-time with an intuitionistic theory based on combinatory logic and safe induction. Preprint, Firenze, 1999. 14 pages.

[17] Cantini, A. Feasible operations and applicative theories based on $\lambda\eta$. *Mathematical Logic Quarterly 46*, 3 (2000), 291–312.

[18] Cantini, A. Polytime, combinatory logic and positive safe induction. *Archive for Mathematical Logic 41*, 2 (2002), 169–189.

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

[20] Cobham, A. The intrinsic computational difficulty of functions. In *Logic, Methodology and Philosophy of Science II.* North Holland, Amsterdam, 1965, pp. 24–30.

[21] Cook, S. A. Feasibly constructive proofs and the propositional calculus. In *Proceedings of the Seventh ACM Symposium on the Theory of Computing.* Association for Computing Machinery, New York, 1975, pp. 83–97.

[22] Cook, S. A., and Kapron, B. M. Characterizations of the basic feasible functionals of finite type. In *Feasible Mathematics*, S. R. Buss and P. J. Scott, Eds. Birkhäuser, Basel, 1990, pp. 71–95.

[23] Cook, S. A., and Urquhart, A. Functional interpretations of feasibly constructive arithmetic. *Annals of Pure and Applied Logic 63*, 2 (1993), 103–200.

[24] Feferman, S. A language and axioms for explicit mathematics. In *Algebra and Logic*, J. Crossley, Ed., vol. 450 of *Lecture Notes in Mathematics.* Springer, Berlin, 1975, pp. 87–139.

[25] Feferman, S. Recursion theory and set theory: a marriage of convenience. In *Generalized recursion theory II, Oslo 1977*, J. E. Fenstad, R. O. Gandy, and G. E. Sacks, Eds., vol. 94 of *Stud. Logic Found. Math.* North Holland, Amsterdam, 1978, pp. 55–98.

[26] Feferman, S. Constructive theories of functions and classes. In *Logic Colloquium '78*, M. Boffa, D. van Dalen, and K. McAloon, Eds. North Holland, Amsterdam, 1979, pp. 159–224.

[27] Feferman, S. Polymorphic typed lambda-calculi in a type-free axiomatic framework. In *Logic and Computation*, W. Sieg, Ed., vol. 106 of *Contemporary Mathematics.* American Mathematical Society, Providence, Rhode Island, 1990, pp. 101–136.

[28] Feferman, S. Logics for termination and correctness of functional programs. In *Logic from Computer Science*, Y. N. Moschovakis, Ed., vol. 21 of *MSRI Publications*. Springer, Berlin, 1991, pp. 95–127.

[29] Feferman, S. Logics for termination and correctness of functional programs II: Logics of strength PRA. In *Proof Theory*, P. Aczel, H. Simmons, and S. S. Wainer, Eds. Cambridge University Press, Cambridge, 1992, pp. 195–225.

[30] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part I. *Annals of Pure and Applied Logic 65*, 3 (1993), 243–263.

[31] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part II. *Annals of Pure and Applied Logic 79*, 1 (1996), 37–52.

[32] Ferreira, F. *Polynomial Time Computable Arithmetic and Conservative Extensions.* PhD thesis, Pennsylvania State University, 1988.

[33] Ferreira, F. Polynomial time computable arithmetic. In *Logic and Computation, Proceedings of a Workshop held at Carnegie Mellon University, 1987*, W. Sieg, Ed., vol. 106 of *Contemporary Mathematics.* American Mathematical Society, Providence, Rhode Island, 1990, pp. 137–156.

[34] Ferreira, F. Stockmeyer induction. In *Feasible Mathematics*, S. Buss and P. Scott, Eds. Birkhäuser, 1990, pp. 161–180.

[35] Girard, J.-Y. *Proof Theory and Logical Complexitiy.* Bibliopolis, Napoli, 1987.

[36] Glass, T., and Strahm, T. Systems of explicit mathematics with non-constructive μ-operator and join. *Annals of Pure and Applied Logic 82* (1996), 193–219.

[37] Hájek, P., and Pudlák, P. *Metamathematics of First-Order Arithmetic.* Perspectives in Mathematical Logic. Springer, 1993.

[38] Hindley, J. R., and Seldin, J. P. *Introduction to Combinators and λ-Calculus.* Cambridge University Press, 1986.

[39] Hofmann, M. Linear types and non-size-increasing polynomial time computation. In *LICS '99, Trento* (1999), IEEE, pp. 464–473.

[40] Hofmann, M. Type systems for polynomial-time computation. Habilitation Thesis, Darmstadt, 1999. Appeared as LFCS Technical Report ECS-LFCS-99-406.

[41] Irwin, R., Kapron, B., and Royer, J. On characterizations of the basic feasible functionals, Part I. *Journal of Functional Programming 11* (2001), 117–153.

[42] Jäger, G. A well-ordering proof for Feferman's theory T_0. *Archiv für mathematische Logik und Grundlagenforschung 23* (1983), 65–77.

[43] Jäger, G., Kahle, R., and Strahm, T. On applicative theories. In *Logic and Foundations of Mathematics*, A. Cantini, E. Casari, and P. Minari, Eds. Kluwer, 1999, pp. 83–92.

[44] Jäger, G., and Pohlers, W. Eine beweistheoretische Untersuchung von $(\Delta^1_2\text{-}\mathsf{CA}) + (\mathsf{BI})$ und verwandter Systeme. In *Sitzungsberichte der Bayerischen Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse.* 1982, pp. 1–28.

[45] Jäger, G., and Strahm, T. Totality in applicative theories. *Annals of Pure and Applied Logic 74*, 2 (1995), 105–120.

[46] Jäger, G., and Strahm, T. Some theories with positive induction of ordinal strength $\varphi\omega 0$. *Journal of Symbolic Logic 61*, 3 (1996), 818–842.

[47] Kapron, B., and Cook, S. A new characterization of type 2 feasibility. *SIAM Journal on Computing 25* (1996), 117–132.

[48] Krajíček, J. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, vol. 60 of *Encyclopedia of Mathematics and its Applications.* Cambridge University Press, 1995.

[49] Leivant, D. A foundational delineation of poly-time. *Information and Computation 110* (1994), 391–420.

[50] Leivant, D. Predicative recurrence in finite type. In *Logical Foundations of Computer Science*, A. Nerode and Y. Matiyasevich, Eds., vol. 813 of *Lecture Notes in Computer Science.* Springer, 1994, pp. 227–239.

[51] Leivant, D. Ramified recurrence and computational complexity I: Word recurrence and poly-time. In *Feasible Mathematics II*, P. Clote and J. Remmel, Eds. Birkhäuser, 1994, pp. 320–343.

[52] Marzetta, M., and Strahm, T. The μ quantification operator in explicit mathematics with universes and iterated fixed point theories with ordinals. *Archive for Mathematical Logic 37*, 5+6 (1998), 391–413.

[53] Melhorn, K. Polynomial and abstract subrecursive classes. *Journal of Computer and System Science 12* (1976), 147–178.

[54] Parsons, C. On a number theoretic choice schema and its relation to induction. In *Intuitionism and Proof Theory, Proceedings of the Summer Conference at Buffalo N.Y., 1968*, J. Myhill, A. Kino, and R. Vesley, Eds. North Holland, Amsterdam, 1970, pp. 459–473.

[55] Pezzoli, E. On the computational complexity of type 2 functionals. In *Computer Science Logic '97*, vol. 1414 of *Lecture Notes in Computer Science*. Springer, 1998, pp. 373–388.

[56] Ritchie, R. W. Classes of predictably computable functions. *Transactions of the American Mathematical Society 106* (1963), 139–173.

[57] Royer, J. Semantics vs. syntax vs. computations: Machine models for type-2 polynomial-time bounded functionals. *Journal of Computer and System Science 54* (1997), 424–436.

[58] Schlüter, A. An extension of Leivant's characterization of poly-time by predicative arithmetic. Preprint, Stanford Univeristy, 1995. 13 pages.

[59] Seth, A. *Complexity Theory of Higher Type Functionals.* PhD thesis, Tata Institute of Fundamental Research, Bombay, 1994.

[60] Sieg, W. Fragments of arithmetic. *Annals of Pure and Applied Logic*, 28 (1985), 33–71.

[61] Sieg, W. Herbrand analyses. *Archive for Mathematical Logic 30*, 5+6 (1991), 409–441.

[62] Strahm, T. Polynomial time operations in explicit mathematics. *Journal of Symbolic Logic 62*, 2 (1997), 575–594.

[63] Strahm, T. The non-constructive μ operator, fixed point theories with ordinals, and the bar rule. *Annals of Pure and Applied Logic 104*, 1–3 (2000), 305–324.

[64] Strahm, T. *Proof-theoretic Contributions to Explicit Mathematics.* Habilitationsschrift, University of Bern, 2001.

[65] Strahm, T. A proof-theoretic characterization of the basic feasible functionals, April 2002. Preprint, 22 pages.

[66] Studer, T. *Object-oriented programming in explicit mathematics: Towards the mathematics of objects.* PhD thesis, Universität Bern, 2001.

[67] Takeuti, G. A second order version of S_2^i and U_2^1. *Journal of Symbolic Logic 56*, 3 (1991), 1038–1063.

[68] Thompson, D. B. Subrecursiveness: machine independet notions of computability in restricted time and storage. *Mathematical Systems Theory 6* (1972), 3–15.

[69] Troelstra, A., and van Dalen, D. *Constructivism in Mathematics*, vol. I. North-Holland, Amsterdam, 1988.

[70] Troelstra, A., and van Dalen, D. *Constructivism in Mathematics*, vol. II. North Holland, Amsterdam, 1988.

[71] Zambella, D. Notes on polynomially bounded arithmetic. *Journal of Symbolic Logic 61*, 3 (1996), 942–966.