Weak theories of operations and types

Thomas Strahm*

Dedicated to Wolfram Pohlers on his retirement

Abstract

This is a survey paper on various weak systems of Feferman's explicit mathematics and their proof theory. The strength of the systems considered in measured in terms of their provably terminating operations typically belonging to some natural classes of computational time or space complexity.

Keywords: Proof theory, Feferman's explicit mathematics, applicative theories, higher types, types and names, partial truth, feasible operations

1 Introduction

In this article we survey recent results about a proof-theoretic approach to computational complexity via theories of operations and types in the sense of Feferman's explicit mathematics. The latter framework was introduced by Feferman [19, 20, 21] in the early 1970s. Beyond its original aim to provide a basis for Bishop-style constructivism, 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 as well as for studying the proof theory of abstract computations.

It is this latter focus which is most important in the present article. The operational or applicative core of explicit mathematics includes forms of combinatory logic and hence comprises a computationally complete functional language with the full defining power of the untyped lambda calculus. In this sense it is more expressive than standard arithmetical systems.

*Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland. Email: `strahm@iam.unibe.ch`. Homepage: `http://www.iam.unibe.ch/~strahm`.

Apart from *operations or rules*, the second basic entity in explicit mathematics are *types*, which can be thought of as successively generated collections of operations. In addition, and this is essential in the explicit approach, extensional types are represented (or named) by intensional operations, uniformly in their parameters. This interplay of operations and types on the level of representations makes explicit mathematics extremely powerful.

As an alternative means of enhancing the first order part of explicit mathematics, we will also consider extensions of applicative theories by a partial truth predicate, leading to an expressive language embodying naive set theory. In this connection, we will review work done by Cantini.

Let us briefly outline the content of the paper. We omit references and credits and refer the reader to the corresponding sections of the paper.

We start off in Section 2 by introducing the first order applicative framework being based on the logic of partial terms. We define the basic theory of operations and words B and introduce two bounded induction schemas on the binary words.

In Section 3 we provide a review of function algebra characterizations of complexity classes and introduce four bounded applicative systems, PT, PTLS, PS, and LS, whose provably total functions coincide with the functions computable in polynomial time, simultaneously polynomial time and linear space, polynomial space, and linear space. We briefly address the lower and upper bound arguments for these systems. In particular, we outline a specific combination of partial cut elimination and a realizability interpretation.

Section 4 addresses higher type issues of the first-order system PT. It is a distinguished advantage of applicative theories that they allow for a very intrinsic and direct discussion of higher type aspects, since higher types arise naturally in the untyped setting. It makes perfect sense to consider the class of higher type functionals which are provably total in a given applicative system. We will discuss the relationship between PT and the Melhorn-Cook-Urquhart basic feasible functionals BFF.

In Section 5 we introduce a theory PET of polynomial time operations with explicit and variable types which is formulated in the full language of explicit mathematics and embodies a weak form of elementary type comprehension. The provably total operations of PET are still the polynomial time computable functions on binary words. We will also consider various extensions of PET by choice, quantification, uniformity and join principles.

In Section 6 we review work by Cantini on extensions of weak applicative theories by forms of self-referential truth with choice and uniformity, which has been essential in obtaining results about corresponding extensions of the system PET.

Finally, in Section 7 we address self-applicative systems proposed by Cantini and Calamai in the realm of so-called implicit computational complexity in the sense of Bellantoni, Cook and Leivant. It turns out that forms of safe induction formulated in a modal language provide very natural applicative characterizations of the functions computable in polynomial time and polynomial space.

We conclude this article by some comments on the relationship between primitive recursion and positive induction.

2 The axiomatic framework

In this section we first describe the informal setting of applicative systems and briefly motivate their underlying logic of partial terms. Then we outline the basic applicative language and theory of operations and words and mention some of its basic consequences and models. We conclude this section by specifying two important induction principles.

2.1 The informal applicative setting

Let us assume that we are given an untyped universe of operations or rules, which can be freely applied to each other. Self-application is meaningful, though not necessarily total. The computational engine of these rules is given by a partial combinatory algebra, featuring partial versions of Curry's combinators k and s. In addition, there is a ground "urelement" structure of the binary words or strings with certain natural operations on them.

Let $\mathbb{W}$ denote the set of (finite) binary words. We will consider the following operations on $\mathbb{W}$:

- s_0 and s_1: binary successors on $\mathbb{W}$ with predecessor p_W
- s_ℓ: (unary) lexicographic successor on $\mathbb{W}$ with predecessor p_ℓ
- $*$: word concatenation
- $\times$: word multiplication

Here s_ℓ denotes the successor in the ordering $<_\ell$ which orders words by length and words of the same length lexicographically. Moreover, $x \times y$ signifies the length of y fold concatenation of x with itself.

2.2 The logic of partial terms

All our theories considered in this survey are based on the classical logic of partial terms (LPT) due to Beeson and Feferman. It is is a modification of first-order predicate logic taking into account partial functions, cf. Beeson [1, 2] and Troelstra and van Dalen [52] for details. It is assumed that variables range over defined objects only. (Composed) terms do not necessarily denote and $t\downarrow$ signifies that t has a value or t denotes. The usual quantifier axioms of predicate logic are modified, e.g. we have

$$A(t) \wedge t\downarrow \;\rightarrow\; (\exists x)A(x)$$

Moreover, strictness axioms claim that subterms of a defined term are defined and that terms occurring in true positive atoms are defined.

For an excellent survey of logics of definedness the reader is referred to Feferman [22]. Feferman distinguished between logics of existence and logics of partial terms in the above-explained sense, whereas the former were pioneered by Scott [41]. On the other hand, the pseudo-applicative terms used in Feferman [19, 21] may be considered as precursors to the logic of partial terms.

2.3 The basic applicative language

Our basic language L is a first order language for the logic of partial terms which includes:

- variables $a, b, c, x, y, z, u, v, f, g, h, \dots$
- constants k, s, p, $\mathsf{p_0}$, $\mathsf{p_1}$, $\mathsf{d_W}$, ϵ, $\mathsf{s_0}$, $\mathsf{s_1}$, $\mathsf{p_W}$, $\mathsf{s_\ell}$, $\mathsf{p_\ell}$, $\mathsf{c_\subseteq}$, $\mathsf{l_W}$, $\dots$
- relation symbols $=$ (equality), $\downarrow$ (definedness), W (binary words)
- arbitrary term application $\circ$

The meaning of the constants will become clear in the next paragraph.

The terms $(r, s, t, \dots)$ and formulas $(A, B, C, \dots)$ of L are defined in the expected manner. We assume the following standard abbreviations and syntactical conventions:

$$\begin{aligned}
t_1 t_2 \dots t_n &:= (\dots(t_1 \circ t_2) \circ \dots \circ t_n) \\
t_1 \simeq t_2 &:= t_1\downarrow \vee\, t_2\downarrow \;\rightarrow\; t_1 = t_2 \\
t \in \mathsf{W} &:= \mathsf{W}(t) \\
t : \mathsf{W}^k \rightarrow \mathsf{W} &:= (\forall x_1 \dots x_k \in \mathsf{W}) t x_1 \dots x_k \in \mathsf{W} \\
t : \mathsf{W}^{\mathsf{W}} \times \mathsf{W} \rightarrow \mathsf{W} &:= (\forall f \in \mathsf{W} \rightarrow \mathsf{W})(\forall x \in \mathsf{W}) t f x \in \mathsf{W}
\end{aligned}$$

Finally, let us write $\overline{w}$ for the canonical closed L term denoting the binary word $w \in \mathbb{W}$.

2.4 The basic theory of operations and words B

The applicative base theory B has been introduced in Strahm [47, 48]. Its logic is the *classical* logic of partial terms. The non-logical axioms of B include:

- partial combinatory algebra:

$$\mathsf{k}xy = x, \qquad \mathsf{s}xy\downarrow \wedge \mathsf{s}xyz \simeq xz(yz)$$

- pairing p with projections p_0 and p_1
- defining axioms for the binary words W with ϵ, the successors s_0, s_1, s_ℓ an the predecessor p_W and and p_ℓ
- definition by cases d_W on W
- initial subword relation $\mathsf{c}_\subseteq$, tally length of words l_W

These axioms are fully spelled out in [47, 48].

Let us turn to the crucial consequences of the axioms about a partial combinatory algebra. For proofs of these standard results, the reader is referred to Beeson [1] or Feferman [19].

Lemma 1 (Explicit definitions and fixed points)

1. *For each* L *term t there exists an* L *term* $(\lambda x.t)$ *so that*

$$\mathsf{B} \vdash (\lambda x.t)\downarrow \ \wedge \ (\lambda x.t)x \simeq t$$

2. *There is a closed* L *term* fix *so that*

$$\mathsf{B} \vdash \mathsf{fix}g\downarrow \ \wedge \ \mathsf{fix}gx \simeq g(\mathsf{fix}g)x$$

Let us quickly remind the reader of two standard models of B, namely the recursion-theoretic model *PRO* and the term model $\mathcal{M}(\lambda\eta)$. For a extensive discussion of many more models of the applicative basis, the reader is referred to Beeson [1] and Troelstra and van Dalen [53].

Example 2 (Recursion-theoretic model *PRO*) Take the universe of binary words and interpret application $\circ$ as partial recursive function application in the sense of ordinary recursion theory.

Example 3 (The open term model $\mathcal{M}(\lambda\eta)$) Take the universe of open λ terms and consider the usual reduction of the extensional untyped lambda calculus $\lambda\eta$, augmented by suitable reduction rules for the constants other than k and s. Interpret application as juxtaposition. Two terms are equal if they have a common reduct and W denotes those terms that reduce to a "standard" word $\overline{w}$.

2.5 Natural induction principles

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 call an 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. We let Pos stand for the collection of positive formulas. Further, an 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 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 note 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.

Below we will distinguish usual notation induction on binary words and the corresponding "slow" induction principle with respect to the lexicographic successor s_ℓ.

$\Sigma^{\mathsf{b}}_{\mathsf{W}}$ notation induction on W:

For each $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula $A(x) \equiv (\exists y \leq fx)B(f,x,y)$,

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

$\Sigma^{\mathsf{b}}_{\mathsf{W}}$ lexicographic induction on W:

For each $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ formula $A(x) \equiv (\exists y \leq fx)B(f,x,y)$,

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

It is now easy, by making use of the fixed point theorem and $\Sigma^{\mathsf{b}}_{\mathsf{W}}$ notation induction on W, to show the existence of a type two functional for bounded recursion on notation, provably in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$. This is the content of the following lemma whose detailed proof can be found in Strahm [48].

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

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

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

Here $t \mid s$ is t if $t \leq s$ and s otherwise.

Similarly, bounded lexicographic recursion is derivable in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$, see Strahm [48] for details.

3 Characterizing complexity classes

We now turn to the characterization of complexity classes by means of our applicative systems. We start our discussion by reviewing some function algebra characterizations of complexity classes and then propose four applicative systems, PT, PTLS, PS, and LS, whose provably total functions coincide with the functions computable in *polynomial time*, *simultaneously polynomial time and linear space*, *polynomial space*, and *linear space*. We sketch lower and upper bounds for these proof-theoretic characterizations.

3.1 Four function algebras

In this subsection we review know recursion-theoretic characterizations of various classes of computational complexity. 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 [15].

In the following we use the notation of Clote [15] 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 *bounded recursion on notation* BRN and *bounded lexicographic recursion* BRL, cf. Strahm [48] for details. A further operator is the *composition operator* COMP. 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 [16]. The delineations of FPtimeLinspace and FPspace are due to Thompson [51]. Finally, the fourth assertion of our theorem is due to Ritchie [38]. For a uniform presentation of all these results we urge the reader to consult Clote [15].

Theorem 5 *We have the following function algebra characterizations of the complexity classes mentioned above:*

1. $[\epsilon, \mathsf{I}, \mathsf{s_N}, \mathsf{s_1}, *, \times; \mathsf{COMP}, \mathsf{BRN}] = \mathrm{FP}\textsc{time}$.
2. $[\epsilon, \mathsf{I}, \mathsf{s_N}, \mathsf{s_1}, *; \mathsf{COMP}, \mathsf{BRN}] = \mathrm{FP}\textsc{time}\mathrm{L}\textsc{inspace}$.
3. $[\epsilon, \mathsf{I}, \mathsf{s}_\ell, *, \times; \mathsf{COMP}, \mathsf{BRL}] = \mathrm{FP}\textsc{space}$.
4. $[\epsilon, \mathsf{I}, \mathsf{s}_\ell, *; \mathsf{COMP}, \mathsf{BRL}] = \mathrm{FL}\textsc{inspace}$.

We now turn to the proof-theoretic characterization of the above four complexity classes by means of suitable applicative theories.

3.2 Provably total functions

Let us first start with a formal definition of the notion of *provably total function* of a given L theory.

Definition 6 A function $F : \mathbb{W}^n \to \mathbb{W}$ is called *provably total in an* L *theory* T, if there exists a closed L term t_F such that

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

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

In the sequel, let $\boldsymbol{\tau}(\mathsf{T}) = \{F : F \text{ provably total in } \mathsf{T}\}$.

3.3 Four applicative systems

In the following we write $\mathsf{B}(*)$ for the extension of B by the obvious axioms about word concatenation on W, namely the standard recursive defining equations and the totality of $*$ on W. We assume that $*$ is a new constant of our applicative language L. Similarly, $\mathsf{B}(*, \times)$ extends $\mathsf{B}(*)$ by the standard axioms about word multiplication. For details, see Strahm [48].

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:

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

We note that a preliminary, more restrictive version of the system PT has previously been studied in Strahm [46] and Cantini [12].

In the sequel let us briefly sketch the lower and upper bound arguments for our applicative systems, which are worked out in full detail in Strahm [48].

3.4 Lower bounds

The lower bounds for our four applicative systems directly follow from Theorem 5 and the crucial Lemma 4, respectively its variant for bounded lexicographic recursion.

Theorem 7 *We have the following lower bounds:*

1. FPtime $\subseteq \boldsymbol{\tau}(\mathsf{PT})$.

2. FPtimeLinspace $\subseteq \boldsymbol{\tau}(\mathsf{PTLS})$.

3. FPspace $\subseteq \boldsymbol{\tau}(\mathsf{PS})$.

4. FLinspace $\subseteq \boldsymbol{\tau}(\mathsf{LS})$.

Let us close this paragraph with the following remarks:

Remarks 8

1. Ferreira's system PTCA^+ ([24, 25]) is directly contained in PT, where PTCA^+ corresponds to Buss' S^1_2 ([5]).
2. The Melhorn-Cook-Urquhart basic feasible functionals resp. the system PV^ω ([18]) are directly contained in PT (see Section 4).

3.5 Partial cut elimination

In order to extract computational content from proofs, we need a sequent-style reformulation of our systems and a preparatory partial cut-elimination result. It 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. Moreover, we will establish upper bounds directly for an extension of our systems by the axioms of *totality of application* and *extensionality of operations*:

Totality of application:

(**Tot**) $$(\forall x)(\forall y)(xy\downarrow)$$

Extensionality of operations:

(**Ext**) $$(\forall f)(\forall g)[(\forall x)(fx \simeq gx) \rightarrow f = g]$$

Observe that in the presence of the totality axiom, the logic of partial terms reduces to ordinary classical predicate logic. 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.

In the following we let $\Gamma, \Delta, \Lambda, \ldots$ range over finite *sequences* of formulas; 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 $(A_1 \wedge \cdots \wedge A_n) \rightarrow (B_1 \vee \cdots \vee B_m)$.

It is now a matter of routine to spell out a sequent-style version of our four applicative systems so that all the main formulas of axioms and rules are *positive*. Hence, partial cut-elimination is applicable in order to show that cuts can be restricted to positive formulas. In the following we write $\mathsf{T}^+ \underset{\star}{\vdash} \Gamma \Rightarrow \Delta$ to express that $\Gamma \Rightarrow \Delta$ has a proof in T^+ where all cut formulas are positive.

3.6 Realizability

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.

The notion of realizability as well as the style and spirit of our realizability theorems are related to the work of Leivant [31], Schlüter [40], and Cantini [13], 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 [13, 40], 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 5; 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 [5, 6, 7]) has been employed with great success in a variety of contexts.

We are now ready to turn to realizability. Our realizers $\rho, \sigma, \tau, \dots$ 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 for positive formulas only.

Definition 9 The notion $\rho \ \mathbf{r} \ A$ ("ρ realizes A") for $\rho \in \mathbb{W}$ and A a positive formula, is given inductively in the following manner:

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

If Δ denotes a sequence $A_1, \ldots, A_n$, then $\rho \ \mathbf{r} \ \Delta$ iff $\rho = \langle i_2, \rho_0 \rangle$ for some $1 \leq i \leq n$ and $\rho_0 \ \mathbf{r} \ A_i$.

The next main lemma about the realizability of quasi-normal PT^+ derivations immediately entails that the provably total functions of PT^+ are computable in polynomial time. The lemma is proved in all detail in Strahm [48].

In the formulation of the lemma, we need the following notation. For an 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 L formulas.

Lemma 10 (Realizability for PT^+) *Let $\Gamma \Rightarrow \Delta$ be a sequent of formulas in Pos with $\Gamma = A_1, \ldots, A_n$ and assume that $\mathsf{PT}^+ \vdash_{\star} \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. 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 $\rho_1, \ldots, \rho_n \in \mathbb{W}$:*

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

Analogous realizability results hold for the systems PTLS, PS, and LS, cf. [48] for details.

3.7 The main theorem concluded

We are now able to piece together the results of Sections 3.4, 3.5 and 3.6 and obtain the following main theorem.

Theorem 11 *We have the following characterizations:*

1. $\boldsymbol{\tau}(\mathsf{PT}) =$ FPtime.
2. $\boldsymbol{\tau}(\mathsf{PTLS}) =$ FPtimeLinspace.
3. $\boldsymbol{\tau}(\mathsf{PS}) =$ FPspace.
4. $\boldsymbol{\tau}(\mathsf{LS}) =$ FLinspace.

In the next section we turn to some higher types aspects of the system PT.

4 Higher type issues

In the last two decades 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 [34]. More than ten years later in 1989, Cook and Urquhart [18] 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 [17, 30], Irwin, Kapron and Royer [27], Pezzoli [37], Royer [39], and Seth [42].

In the following let us briefly discuss the relationship of PV^ω with our first-order applicative theory PT.

4.1 Higher types in the language L

The collection $\mathcal{T}$ *of finite type symbols* $(\alpha, \beta, \gamma, \ldots)$ 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 $(\alpha_1 \to (\alpha_2 \to \cdots \to (\alpha_{k-1} \to \alpha_k) \cdots))$.

The abstract *intensional type structure* $\langle(\mathsf{IT}_\alpha, =)\rangle_{\alpha \in \mathcal{T}}$ in the applicative language L is now given by inductively defining the formula IT_α as follows:

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

Equality in IT_α is simply the restriction of equality in PT. Alternatively, one can consider an extensional type structure, cf. [53, 48].

4.2 The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. PV^ω includes:

- the simply typed lambda calculus over the base type of binary words
- basic operations on words, essentially the base operations of PT
- a type two functional for bounded recursion on notation
- notation induction on binary words for Σ^b_1 or *NP* formulas

For an exact definition, cf. e.g. Strahm [48]. We observe that due to Lemma 4, we indeed have a type two functional for bounded recursion on notation which has the correct type, provably in PT. Using the intensional type structure $\langle(\mathsf{IT}_\alpha, =)\rangle_{\alpha\in\mathcal{T}}$ sketched above, it is then a matter of routine to check that PV^ω can be directly interpreted in PT. This shows that the basic feasible functionals in all finite types are provably total in PT.

The question arises whether indeed the BFFs are *exactly* the provably total functionals of PT. This question has been answered in the positive for the *type two* BFFs in Strahm [49] by using an extension of the realizability argument sketched above. Moreover, it follows from the work in Cantini [14] that this result holds with respect to arbitrary finite types if one considers an intuitionistic version of PT. Therefore we can summarize:

Theorem 12 *1. The system* PV^ω *is contained in* PT*; i.e., the basic feasible functionals in all finite types are provably total in* PT*.*

2. The provably total type 2 functionals of PT *coincide exactly with the basic feasible functionals of type 2.*

Let us conclude this section with the following conjecture.

Conjecture 13 *The classical theory* PT *characterizes the basic feasible functionals in all finite types.*

5 Adding types and names

In this section, we will describe PET, a theory of **p**olynomial time operations with **e**xplicit **t**ypes. The theory PET is an extension of the applicative base theory $\mathsf{B}(*, \times)$ by means of a natural restriction of elementary comprehension, which is one of the crucial principles of explicit mathematics, see Feferman [19, 21]. Below we will use the language of explicit mathematics due to Jäger [28] which is based on a so-called naming relation $\Re$. The type existence axioms are naturally presented by means of a finite axiomatisation in the spirit of Feferman and Jäger [23]. The theory PET has been introduced in Spescha and Strahm [44].

5.1 The informal setting of types and names

Types in explicit mathematics are collections of operations and must be thought of as being generated successively from preceding ones. In contrast to the restricted character of operations, types can have quite complicated defining properties. What is essential in the whole explicit mathematics approach, however, is the fact that types are again represented by operations or, as we will call them in this case, *names.* Thus each type U is named or represented by a name u; in general, U may have many different names or representations. It is exactly this interplay between operations and types on the level of names which makes explicit mathematics extremely powerful and, in fact, witnesses its explicit character.

Types are extensional and have (explicit) names which are intensional. The names are generated via uniform operations and the link to the types they are referring to is established by the naming relation $\Re$. The element relation $\in$ is also a relation between an individual and a type, expressing that the individual is a member of the type. The formalization of explicit mathematics using a naming relation $\Re$ is due to Jäger [28].

5.2 The language of types and names

The language $\mathbb{L}$ is a two-sorted language extending L by

- type variables $U, V, W, X, Y, Z, \dots$
- binary relation symbols $\Re$ (naming) and $\in$ (elementhood)
- new (individual) constants w (initial segment of W), id (identity), dom (domain), un (union), int (intersection), and inv (inverse image)

The *formulas* $(A, B, C, \dots)$ of $\mathbb{L}$ are built from the atomic formulas of L as well as formulas of the form

$$(s \in X), \quad \Re(s, X), \quad (X = Y)$$

by closing under the boolean connectives and quantification in both sorts. The formula $\Re(s, X)$ reads as "the individual s is a name of (or represents) the type X".

We use the following abbreviations:

$$\begin{aligned} \Re(s) &:= (\exists X)\Re(s, X), \\ s \,\dot{\in}\, t &:= (\exists X)(\Re(t, X) \wedge s \in X). \end{aligned}$$

5.3 The theory PET

The following axioms state that each type has a name, that there are no homonyms and that equality of types is extensional.

Ontological axioms:

(O1) $(\exists x)\Re(x, X)$

(O2) $\Re(a, X) \wedge \Re(a, Y) \rightarrow X = Y$

(O3) $(\forall z)(z \in X \leftrightarrow z \in Y) \rightarrow X = Y$

In the sequel we let $\mathsf{W}_a(x)$ stand for $\mathsf{W}(x) \wedge x \leq a$. The following axioms provide a finite axiomatization of a restricted form of the schema of elementary comprehension.

Type existence axioms:

($\mathbf{w}_a$) $a \in \mathsf{W} \rightarrow \Re(\mathsf{w}(a)) \wedge (\forall x)(x \dot{\in} \mathsf{w}(a) \leftrightarrow \mathsf{W}_a(x))$

(**id**) $\Re(\mathsf{id}) \wedge (\forall x)(x \dot{\in} \mathsf{id} \leftrightarrow (\exists y)(x = (y, y)))$

(**inv**) $\Re(a) \rightarrow \Re(\mathsf{inv}(f, a)) \wedge (\forall x)(x \dot{\in} \mathsf{inv}(f, a) \leftrightarrow fx \dot{\in} a)$

(**un**) $\Re(a) \wedge \Re(b) \rightarrow \Re(\mathsf{un}(a, b)) \wedge (\forall x)(x \dot{\in} \mathsf{un}(a, b) \leftrightarrow (x \dot{\in} a \vee x \dot{\in} b))$

(**int**) $\Re(a) \wedge \Re(b) \rightarrow \Re(\mathsf{int}(a, b)) \wedge (\forall x)(x \dot{\in} \mathsf{int}(a, b) \leftrightarrow (x \dot{\in} a \wedge x \dot{\in} b))$

(**dm**) $\Re(a) \rightarrow \Re(\mathsf{dom}(a)) \wedge (\forall x)(x \dot{\in} \mathsf{dom}(a) \leftrightarrow (\exists y)((x, y) \dot{\in} a))$

In contrast to the usual formulation of elementary comprehension in explicit mathematics (cf. e.g. Feferman and Jäger [23]), we do not claim that the collection of binary words forms a type, but merely that for each word a, the collection $\{x \in \mathsf{W} : x \leq a\}$ forms a type, uniformly in a. In addition, there are no complement types. The remaining type existence axioms are identical to the ones in [23].

Finally, the principle of type induction along W reads in the expected manner.

Type induction on W:

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

The theory PET is defined to be the extension of the first-order applicative theory $\mathsf{B}(*, \times)$ by

- the ontological axioms

- the above type existence axioms
- type induction on W

In Spescha and Strahm [44] it is shown that the finite axiomatisation of type existence in PET gives rise to a natural restriction of the well-known schema of elementary comprehension in explicit mathematics.

5.4 The proof-theoretic strength of PET

Let PT^- be PT without universal quantifiers in induction formulas. Clearly, PT^- proves the totality of the polynomial time computable functions, since it is strong enough to represent bounded recursion on notation in the form of a type two functional (cf. Lemma 4). Indeed, PET is a conservative extension of PT^- as is shown in Spescha and Strahm [44].

Theorem 14 *We have the following proof-theoretic results:*

1. PET *is a conservative extension of* PT^-.
2. $\boldsymbol{\tau}(\mathsf{PT}^-) = \text{FP}\textsc{time}$.

The lower bound uses a rather involved embedding of PT^- into PET. The interpretation uses a bootstrapping functional mapping each operation f on W to an operation f^* such that $f^*x = \max\limits_{y \subseteq x} fy$.

For the proof of the upper bound one starts off from a model of PT^- and extends it to a model of PET satisfying the same first order sentences. The construction is carried out in stages by defining the set of names and their extensions successively. Then one can show that the so-obtained model enjoys type induction.

For full details of these arguments, see Spescha and Strahm [44].

5.5 Extensions of PET

In addition to the principles (**Tot**) and (**Ext**) discussed above, Cantini [14] has considered a form of positive choice in the context of PT with a partial truth predicate (cf. Section 6) and shows that this principle does not increase the proof-theoretic strength. Cantini's result can be used to show that the following form of the axiom of choice formulated in the language $\mathbb{L}$ does not increase the strength of PET.

Positive axiom of choice:

(**AC**) $\quad (\forall x \in \mathsf{W})(\exists y \in \mathsf{W})A(x,y) \rightarrow (\exists f : \mathsf{W} \rightarrow \mathsf{W})(\forall x \in \mathsf{W})A(x, fx)$

where $A(x,y)$ is a positive elementary formula.

Cantini has also shown in [14] that adding a uniformity principle for positive formulas of L yields an extension of PT whose provably total functions are still the functions computable in polynomial time. In our context, we can state Cantini's principle as follows.

Positive uniformity principle:

(**UP**) $\qquad (\forall x)(\exists y \in \mathsf{W})A(x,y) \to (\exists y \in \mathsf{W})(\forall x)A(x,y)$

where $A(x,y)$ is positive elementary.

The principle (**UP**) leads to a very natural extension of PET by adding a type existence axiom for universal quantification; this axiom is the natural dual analogue of the domain type present in PET.

Universal quantification:

(**all**) $\qquad \Re(a) \to \Re(\mathsf{all}(a)) \wedge (\forall x)(x \,\dot{\in}\, \mathsf{all}(a) \leftrightarrow (\forall y)((x,y) \,\dot{\in}\, a))$

The presence of the axiom (**all**) makes the type existence axioms more symmetric, i.e. the types are generated from base types (initial segments of W and the identity type) by closing under domains, unions, intersections, existential quantification (inverse image) and universal quantification.

In order to see that (**all**) does not increase the proof-theoretic strength of PET, one shows that any model of $\mathsf{PT} + (\mathbf{UP})$ can be extended to a a model of $\mathsf{PET} + (\mathbf{all})$. The presence of (**UP**) is pivotal in the treatment of (**all**). For a complete exposition of these results, see Spescha and Strahm [44].

Theorem 15 *The provably total functions of* PET *augmented by any combination of the principles* (**all**)*,* (**UP**)*,* (**AC**)*,* (**Tot**)*, and* (**Ext**) *coincide with the polynomial time computable functions.*

The next natural step is to add the so-called *Join axiom*, which constructs disjoint unions of types named by an operation; it has been widely studied for many systems of explicit mathematics. The Join axioms are given by the following assertions (**J.1**) and (**J.2**) (j denotes a new constant).

Join axioms:

(**J.1**) $\qquad \Re(a) \wedge (\forall x \,\dot{\in}\, a)\Re(fx) \to \Re(\mathsf{j}(a,f))$

(**J.2**) $\qquad \Re(a) \wedge (\forall x \,\dot{\in}\, a)\Re(fx) \to (\forall x)(x \,\dot{\in}\, \mathsf{j}(a,f) \leftrightarrow \Sigma(f,a,x))$

where $\Sigma(f,a,x)$ is the formula

$$(\exists y)(\exists z)(x = (y,z) \wedge y \,\dot{\in}\, a \wedge z \,\dot{\in}\, fy)$$

In Spescha [43] and Spescha and Strahm [45] the realizability interpretation of the first order language L is extended to the language of types and names $\mathbb{L}$. In combination with a partial cut elimination argument, it is possible to show that the *intuitionistic* version of PET plus the Join axioms can be realized using polynomial time computable functions. Currently, work is underway in order to extend this result to classical logic.

6 Partial truth

In this section we address some interesting extensions of PT^+ which have been proposed and studied by Cantini [14]. The idea is to augment PT^+ by

- a (form of) self-referential truth (à la Aczel, Feferman, Kripke, etc.), providing a fixed point theorem for predicates
- an axiom of choice for operations and a uniformity principle, restricted to positive conditions

These extensions do not alter the proof-theoretic strength of PT, a fact that has been heavily used in the previous section in studying extensions of our theory PET.

In the following let us briefly report on some of the many results obtained in Cantini [14]. For a thorough exposition of frameworks for truth and abstraction based on combinatory logic, cf. Cantini [10] and Kahle [29].

6.1 The language L_{T}

The (first order) language L_{T} is an extension of the language L by

- a new unary predicate symbol T for *truth*
- new individual constants $\dot{=}$, $\dot{\mathsf{W}}$, $\dot{\wedge}$, $\dot{\vee}$, $\dot{\forall}$, $\dot{\exists}$

For each positive formula A of L_{T} we can inductively define a term $[A]$ whose free variables are exactly the free variables of A:

$$\begin{array}{rcl}
[t = s] & := & (\dot{=}ts) \\
[\mathsf{T}(t)] & := & t \\
[s \in \mathsf{W}] & := & \dot{\mathsf{W}}s \\
[A \wedge B] & := & \dot{\wedge}[A][B] \\
[A \vee B] & := & \dot{\vee}[A][B] \\
[(\forall x)A] & := & \dot{\forall}(\lambda x.[A]) \\
[(\exists x)A] & := & \dot{\exists}(\lambda x.[A])
\end{array}$$

We have that $\lambda x.[A]$ can be interpreted as the propositional function defined by the formula A. We can now interpret the language of naive set theory by defining $x \in a$ as $\mathsf{T}(ax)$ and understand $\{x : A\}$ as $\lambda x.[A]$.

6.2 The truth axioms

The truth axioms for the positive fragment of L_{T} spell out the expected clauses according to the reductionist semantics as follows:

Truth axioms:

$$\begin{array}{rcl}
\mathsf{T}(\dot{=}xy) & \leftrightarrow & x = y \\
\mathsf{T}(\dot{\mathsf{W}}x) & \leftrightarrow & \mathsf{W}(x) \\
\mathsf{T}(x \dot{\wedge} y) & \leftrightarrow & \mathsf{T}(x) \wedge \mathsf{T}(y) \\
\mathsf{T}(x \dot{\vee} y) & \leftrightarrow & \mathsf{T}(x) \vee \mathsf{T}(y) \\
\mathsf{T}(\dot{\forall} f) & \leftrightarrow & (\forall x)\mathsf{T}(fx) \\
\mathsf{T}(\dot{\exists} f) & \leftrightarrow & (\exists x)\mathsf{T}(fx)
\end{array}$$

One of the many interesting consequences of these axioms is a second recursion or fixed point theorem for positive predicates, which can be obtained by lifting the fixed point theorem for combinatory logic (cf. Lemma 1) to the truth-theoretic language, cf. Cantini [10, 14].

6.3 Adding positive choice and uniformity

We can formulate positive choice and uniformity principles in the language L_{T} as follows:

Positive choice and uniformity in L_{T}:

(**AC**) $\quad (\forall x \in \mathsf{W})(\exists y \in \mathsf{W})\mathsf{T}(axy) \to (\exists f : \mathsf{W} \to \mathsf{W})(\forall x \in \mathsf{W})\mathsf{T}(ax(fx))$

(**UP**) $\quad (\forall x)(\exists y \in \mathsf{W})\mathsf{T}(axy) \to (\exists y \in \mathsf{W})(\forall x)\mathsf{T}(axy)$

One of the many results obtained in Cantini [14] is stated in the following theorem. It has been used in Spescha and Strahm [44] in order to show the conservativity of various extensions of PET.

Theorem 16 $\boldsymbol{\tau}(\mathsf{PT}^+ + \mathsf{truth\ axioms} + \mathbf{AC} + \mathbf{UP}) = \mathrm{FP}\textsc{time}$.

The proof methods used by Cantini include a subtle internal forcing semantics, non-standard variants of realizability and partial cut elimination properties. The forcing interpretation is very elegant and makes direct use of the truth predicate T.

7 Safe induction

Apart from the world of bounded recursion schemas, bounded arithmetic and bounded applicative theories there is the realm of so-called *tiered systems* in the sense of Cook and Bellantoni (cf. e.g. [3]) and Leivant (cf. e.g. [31, 32]). 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.

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. Bellantoni, Niggl, Schwichtenberg [4], Hofmann [26], and Leivant [33],

Finally, the tiered approach has provided neat distinctions between slow growing and fast growing proof theories, see e.g. Wainer [54] and Ostrin and Wainer [36].

Below let us briefly address some recent work along the lines of implicit characterizations in the context of untyped applicative theories based on classical logic.

7.1 Polynomial time

In our applicative setting the above-mentioned "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 [13] for such systems.

A more elegant viewpoint of the predicative regime is to consider a modal framework. Extend the language L by a modal operator $\Box$ and let $\Box$ obey the laws of an S4 modality. Let $t \in \Box\mathsf{W}$ stand for $\Box(t \in \mathsf{W})$. Then W and $\Box\mathsf{W}$ play the role of normal and safe strings in the Bellantoni-Cook sense, respectively. We call a formula *positive safe* if it is positive and does not involve the $\Box$ operator. Accordingly, we can formulate the following natural induction principle.

Positive safe notation induction:

For each positive safe formula $A(x)$,

$$A(\epsilon) \wedge (\forall x \in \Box\mathsf{W})(A(x) \rightarrow A(\mathsf{s}_0 x) \wedge A(\mathsf{s}_1 x)) \rightarrow (\forall x \in \Box\mathsf{W})A(x)$$

Let PR^μ denote the extension of the applicative theory B based on the classical modal predicate logic S4 and the schema of positive safe notation induction. The notion of a provably total word function can be suitably adapted for PR^μ, taking into account the two sorts W and $\Box\mathsf{W}$, cf. [13] for details.

We are ready to state the following theorem, which is proved in Cantini [13] by making use of cut elimination and realizability by Cook-Bellantoni functions.

Theorem 17 $\boldsymbol{\tau}(\mathsf{PR}^\mu) = \mathrm{FP}\textsc{time}$.

7.2 Polynomial space

More recently, Calamai and Cantini [9, 8] have proposed an extension of PR^μ, termed PR^μ_p, where induction is strengthened to so-called positive safe tree induction with pointers. The principle is inspired by Oitavem's recent tiered characterization of FPSPACE in [35].

Positive safe tree induction with pointers:
For each positive safe formula $A(x, y)$,

$$
\begin{aligned}
&(\forall p \in \Box\mathsf{W})A(\epsilon, p) \wedge \\
&(\forall x \in \Box\mathsf{W})(\forall p \in \Box\mathsf{W})(A(x, \mathsf{s}_0 p) \wedge A(x, \mathsf{s}_1 p) \to A(\mathsf{s}_0 x, p) \wedge A(\mathsf{s}_1 x, p)) \\
&\qquad \to (\forall x \in \Box\mathsf{W})(\forall p \in \Box\mathsf{W})A(x, p)
\end{aligned}
$$

The proof of the theorem below of Calamai and Cantini makes use of cut elimination and realizability by functions in Oitavem's function algebra with pointers and tree recursion.

Theorem 18 $\boldsymbol{\tau}(\mathsf{PR}^\mu_p) = \mathrm{FP}\textsc{space}$.

8 Conclusion

In this article we have considered a number of applicative theories (with and without types or self-referential truth, with and without modality) whose induction principles are formulated for a suitable subclass of positive formulas.

Regarding induction for *arbitrary positive formulas*, say in the first order language L, one captures exactly the primitive recursive functions. For definiteness, let (Pos-I_W) denote induction on W for formulas in Pos. Then $\boldsymbol{\tau}(\mathsf{B} + (\mathsf{Pos}\text{-}\mathsf{I}_\mathsf{W}))$ coincides with the primitive recursive functions. This result was first established by Cantini in [11] using asymmetric interpretation and formalized semantics in $\mathsf{I}\Sigma_1$ and can be considered as a generalization to the applicative context of the well-known Parsons-Mints-Takeuti theorem. The characterization theorem can also be established by the realizability techniques presented in this article (cf. Cantini [14], Strahm [48]). However, it has to be mentioned that Cantini's original result [11] is even a bit stronger, since negated equations in induction formulas are allowed.

We conclude this article by mentioning that the realizability techniques of this paper have recently been helpful in the context of abstract many sorted algebras with non-computable equality in establishing a further generalization of the Parsons-Mints-Takeuti theorem, cf. Strahm and Zucker [50].

References

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

[2] 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.

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

[4] 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.

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

[6] 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. Westerstrahl, Eds. Elsevier, North Holland, Amsterdam, 1994, pp. 29–68.

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

[8] Calamai, G. *Proof-theoretic contributions to computational complexity.* PhD thesis, University of Siena, 2008.

[9] Cantini, A. A footnote on the Parsons-Mints-Takeuti theorem. Talk at *Recent Trends in Proof Theory*, Bern, July 2008.

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

[11] 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.

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

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

[14] Cantini, A. Choice and uniformity in weak applicative theories. In *Logic Colloquium '01*, M. Baaz, S. Friedman, and J. Krajíček, Eds., vol. 20 of *Lecture Notes in Logic.* Association for Symbolic Logic, 2005.

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

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] FEFERMAN, S. Definedness. *Erkenntnis 43* (1995), 295–320.

[23] 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.

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

[25] 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.

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

[27] IRWIN, R., KAPRON, B., AND ROYER, J. On characterizations of the basic feasible functionals, Part I. *Journal of Functional Programming 11* (2001), 117–153.

[28] JÄGER, G. Induction in the elementary theory of types and names. In *Computer Science Logic '87*, E. Börger, H. Kleine Büning, and M.M. Richter, Eds., vol. 329 of *Lecture Notes in Computer Science.* Springer, Berlin, 1988, pp. 118–128.

[29] KAHLE, R. *The Applicative Realm.* Habilitation Thesis, Tübingen, 2007. Appeared in Textos de Mathemática 40, Departamento de Mathemática da Universidade de Coimbra, Portugal, 2007.

[30] KAPRON, B., AND COOK, S. A new characterization of type 2 feasibility. *SIAM Journal on Computing 25* (1996), 117–132.

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

[32] 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.

[33] Leivant, D. Implicit computational complexity for higher type functionals (Extended abstract). In *CSL '02*, J. Bradfield, Ed., vol. 2471 of *Lecture Notes in Computer Science*. Springer, 2002, pp. 367–381.

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

[35] Oitavem, I. Characterizing PSPACE with pointers. *Mathematical Logic Quarterly 54*, 3 (2008), 323 – 329.

[36] Ostrin, G., and Wainer, S. S. Elementary arithmetic. *Annals of Pure and Applied Logic 133* (2005), 275–292.

[37] 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.

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

[39] 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.

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

[41] Scott, D. S. Identity and existence in intuitionistic logic. In *Applications of Sheaves*, M. P. Fourman, C. J. Mulvey, and D. S. Scott, Eds. Springer, Berlin, 1979. Lecture Notes in Mathematics 753.

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

[43] Spescha, D. *Weak systems of explicit mathematics.* PhD thesis, Universität Bern. In preparation.

[44] Spescha, D., and Strahm, T. Elementary explicit types and polynomial time operations. *Mathematical Logic Quarterly*. To appear.

[45] Spescha, D., and Strahm, T. The join principle in weak systems of explicit mathematics. In preparation.

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

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

[48] Strahm, T. Theories with self-application and computational complexity. *Information and Computation 185* (2003), 263–297.

[49] Strahm, T. A proof-theoretic characterization of the basic feasible functionals. *Theoretical Computer Science 329* (2004), 159–176.

[50] Strahm, T., and Zucker, J. Primitive recursive selection functions for existential assertions over abstract algebras. *Journal of Logic and Algebraic Programming 76*, 2 (2008), 175 – 197.

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

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

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

[54] Wainer, S. S. Provable recursiveness and complexity. In *Logic Colloquium '01*, M. Baaz, S. Friedman, and J. Krajíček, Eds., vol. 20 of *Lecture Notes in Logic*. Association for Symbolic Logic, 2005.

Bern, February 15, 2009