# An applicative theory for FPH

Reinhard Kahle
CENTRIA and DM, FCT, Universidade Nova de Lisboa, P-2829-516 Caparica, Portugal
kahle@mat.uc.pt

Isabel Oitavem
CMAF, Universidade de Lisboa and DM, FCT, Universidade Nova de Lisboa, P-2829-516 Caparica, Portugal
oitavem@fct.unl.pt

In this paper we introduce an applicative theory which characterizes the polynomial hierarchy of time.

## 1 Introduction

In this paper we define an applicative theory whose provably total functions are those which belong to the polynomial hierarchy of time.

Considering theories which characterize classes of computational complexity, there are three different approaches: in one, the functions which can be defined within the theory are "automatically" within a certain complexity class. In such an account, the syntax has to be restricted to guarantee that one stays in the appropriate class. This results, in general, in the problem that certain definitions of functions do not work any longer, even if the function is in the complexity class under consideration. In a second account, the underlying logic is restricted.[1] In the third account, one does not restrict the syntax, allowing, in general, to write down "function terms" for arbitrary (partial recursive) functions, nor the logic, but only for those function terms which belong to the complexity class under consideration, one can *prove* that they have a certain characteristic property, usually, the property that they are "provably total" (see Definition 14 below). While the function terms, according to the underlying syntactical framework, may have a straightforward computational character, i.e., as $\lambda$ terms, the logic which is used to prove the characteristic property may well be *classical*.

Here, we follow the third account, using *applicative theories* as underlying framework.

Applicative theories are the first-order part Feferman's system of explicit mathematics [Fef75, Fef79]. They provide a very handy framework to formalize theories of different strength, including to characterize classes of computational complexity. A first characterization of polynomial time operations in applicative theories was given by Strahm in [Str97]. A uniform approach to varies complexity classes, including FPTIME, FPSPACE, FPTIME-FLINSPACE, and FLINSPACE was given by the same author in his Habilitationsschrift, published in [Str03]. These characterizations are based on bounded schemes in the vein of Cobham [Cob65] (see also [Clo99]). Cantini [Can02] gave, at the same time, a characterization of FPTIME in an applicative framework following the approach of Bellantoni and Cook [BC92] which separates the input positions of functions in normal and safe.

Work partially supported by the ESF research project *Dialogical Foundations of Semantics* within the ESF Eurocores program *LogICCC*, LogICCC/0001/2007 (funded by the Portuguese Science Foundation, FCT). The second author was also supported by the project *Functional interpretations of arithmetic and analysis*, PTDC/MAT/104716/2008 from FCT.

[1] As an example for this approach we may cite [Sch06].

Steffen van Bakel, Stefano Berardi, Ulrich Berger (Eds.):
Classical Logic and Computation 2010 (Cl&C'10)
EPTCS 47, 2011, pp. 44–56, doi:10.4204/EPTCS.47.6

On the base of a characterization of the functions in the Polynomial Hierarchy which uses a monotonicity condition, given in [BALO1x], we present here an applicative theory for FPH. Given a function algebra, the main objective of defining a corresponding theory is, of course, to introduce an adequate induction scheme which allows to prove properties for the functions under consideration. In section 2 we rewrite the input-sorted characterization of FPH given in [BALO1x] as a non-sorted characterization, in Cobham style, by introducing bounds in the recursion schemes. The next sections are concerned with the main goal of this paper: to define an induction scheme which takes care of the monotonicity condition. While the proof of the lower bound follows from a (more or less) straightforward embedding of the function algebra described in section 2, the upper bound is carried out by an adaptation of the proof(s) given by Strahm in [Str03].

Note, that Strahm also treats the polynomial hierarchy in [Str03], but in a quite different way which involves a special type two functional.

**Notation.** We use $\mathbb{W}$ to denote the word algebra generated by $\varepsilon$ (source), and $S_0$ and $S_1$ (successors). $\mathbb{W}$ is usually interpreted over the set of binary words $\{0,1\}^*$. Given $x,y \in \mathbb{W}$, $|x|$ is the length of $x$ and $x|_y$ denotes the word corresponding to the first $|y|$ bits of $x$. $x'$ denotes the numeric successor of $x$, and it defined according to the equations $\varepsilon' = S_0(\varepsilon)$, $(S_0(x))' = S_1(x)$ and $(S_1(x))' = S_0(x')$. The letters $x,y,z,w,\ldots$ denote usually variables, while $f,g,h,s,r,\ldots$ denote function symbols. $\vec{x}$ and $\vec{f}$ denote, respectively, a sequence of variables and functions of the appropriate arity.

## 2 Function algebras for FPH

In this section we work with two function algebras. One formulated in a non-sorted context, and the other formulated in a two-input-sorted context following notation introduced by Bellantoni and Cook in [BC92]. In the sorted context, function arguments have two sorts, *normal* and *safe*. We write them by this order, separated by a semicolon: $f(\vec{x};\vec{y})$.

PH, the polynomial hierarchy of time, is usually defined as $\bigcup_i \Sigma_i$ or $\bigcup_i \Delta_i$ with $\Sigma_0 = \Delta_0 = \mathrm{P}$ and, for $i \geq 0$, $\Sigma_{i+1} = \mathrm{NP}(\Sigma_i)$ and $\Delta_{i+1} = \mathrm{P}(\Sigma_i)$. The corresponding function classes are $\Box_i = \mathrm{FPTIME}(\Delta_i) = \mathrm{FPTIME}(\Sigma_{i-1})$, for $i \geq 1$, and $\mathsf{FPH} = \bigcup_i \Box_i = \mathrm{FPTIME}(\mathrm{PH})$.

Consider the following partial order over $\mathbb{W}$, using $\leq$ as the natural one on $\{0,1\}$.

**Definition 1.** *For $w,v \in \mathbb{W}$, we write $w \preceq v$ if $|w| < |v|$, or $|w| = |v|$ and $\forall i.w_i \leq v_i$. We write $w \prec v$ if $w \preceq v$ but $w \neq v$.*

**Definition 2.**
1. *A function $h$ is called* monotone *if, for all $z \in \mathbb{W}$, $z \preceq h(\vec{x},z)$.*
2. *A two-sorted function $h$, with at least one safe argument, is called* monotone *if, for all $z \in \mathbb{W}$, $z \preceq h(\vec{x};\vec{y},z)$.*

**Definition 3.**
1. *Given a function $h$, its* monotone section *is the function*

$$h^m(\vec{x},z) = \begin{cases} h(\vec{x},z) & \text{if } z \preceq h(\vec{x},z), \\ z & \text{otherwise}. \end{cases}$$

2. *Given a two-sorted function $h$, with at least one safe argument, its* monotone section *is the function*

$$h^m(\vec{x};\vec{y},z) = \begin{cases} h(\vec{x};\vec{y},z) & \text{if } z \preceq h(\vec{x};\vec{y},z), \\ z & \text{otherwise}. \end{cases}$$

Clearly, monotone sections are always monotone functions.

## 2.1 Predicative approach

Consider the class $[\mathscr{B}; PC, PRN, PPR]$ of two-input-sorted functions.

$\mathscr{B}$ is the set of basic functions defined as follows:

1. $\varepsilon$ (a zero-ary function);
2. $\pi_i^{k,n}(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+n}) = x_i$, for each $1 \le i \le k+n$;
3. $S_i(x;) = xi$, $i \in \{0,1\}$;
4. $S_i(z;x) = \begin{cases} xi & \text{if } |x| < |z|, \\ x & \text{otherwise,} \end{cases}$ $i \in \{0,1\}$;
5. $P(;\varepsilon) = \varepsilon$, $P(;xi) = x$, $i \in \{0,1\}$;
6. $p(;\varepsilon) = \varepsilon$, $p(;x') = x$;
7. $Q(;\varepsilon,y,z_0,z_1) = y$, $Q(;xi,y,z_0,z_1) = z_i$, $i \in \{0,1\}$;
8. $\times(x,y;) = 1^{|x|\times|y|}$.

$PC$, $PRN$ and $PPR$ are the following operators:

- Predicative composition: Given $g,\vec{r},\vec{s}$, their predicative composition $f = PC(g,\vec{r},\vec{s})$ is defined by

$$f(\vec{x};\vec{y}) = g(\vec{r}(\vec{x};);\vec{s}(\vec{x};\vec{y})).$$

- Predicative recursion on notation: Given $g,h_0,h_1$, the predicative recursion on notation scheme defines a function $f = PRN(g,h_0,h_1)$ by

$$\begin{aligned} f(\varepsilon,\vec{x};\vec{y}) &= g(\vec{x};\vec{y}), \\ f(zi,\vec{x};\vec{y}) &= h_i(z,\vec{x};\vec{y},f(z,\vec{x};\vec{z})), \qquad i \in \{0,1\} \end{aligned}$$

- Predicative primitive recursion: Given $g$ and $h$, the predicative primitive recursion scheme defines a function $f = PPR(g,h)$ by

$$\begin{aligned} f(\varepsilon,\vec{x};\vec{y}) &= g(\vec{x};\vec{y}), \\ f(z',\vec{x};\vec{y}) &= h(z,\vec{x};\vec{y},f(z,\vec{x};\vec{z})). \end{aligned}$$

**Proposition 4** ([BC92] and [Oit97])**.**

- $[\mathscr{B}; PC, PRN] = \text{FPtime}$,
- $[\mathscr{B}; PC, PRN, PPR] = \text{FPspace}$.

**Definition 5.** *Given g and h, the* predicative monotone primitive recursion scheme *MPPR is defined by* $MPPR(g,h) = PPR(g,h^m)$.

**Proposition 6** ([BALO1x])**.** $[\mathscr{B}; PC, PRN, MPPR] = \text{FPH}$.

**Remark 7.** *For all* $f \in [\mathscr{B}; PC, PRN, PPR]$*:*

1. *there exists a* $F \in [\mathscr{B}; PC, PRN, PPR]$ *such that* $\forall \vec{x},\vec{y}.F(\vec{x},\vec{y};) = f(\vec{x};\vec{y})$*;*
2. *there exists a polynomial* $q_f$ *such that* $\forall \vec{x},\vec{y}.|f(\vec{x};\vec{y})| \le \max\{q_f(|\vec{x}|),\max_i |y_i|\}$.

*This remark holds also if* $[\mathscr{B}; PC, PRN, PPR]$ *is replaced by* $[\mathscr{B}; PC, PRN, MPPR]$.

See [Oit97] for details.

## 2.2 Bounded approach

Consider the class $[\mathscr{I}; C, BRN, BPR]$ where:

- $\mathscr{I}$ is the set of initial functions:
  1. $\varepsilon$,
  2. $S_i(x) = xi, \quad i \in \{0,1\}$,
  3. $\pi^n_j(x_1,\ldots,x_n) = x_j$, $1 \leq j \leq n$,
  4. $Q(\varepsilon,y,z_0,z_1) = y$, $Q(xi,y,z_0,z_1) = z_i, \quad i \in \{0,1\}$,
  5. $\times(x,y) = 1^{|x|\times|y|}$.
- $C$, $BRN$ and $BPR$ are the following operators:
  - Composition: Given $g$ and $\vec{h}$, their composition $f = C(g,\vec{h})$ is given by $f(\vec{x}) = g(\vec{h}(\vec{x}))$,
  - Bounded recursion on notation: Given $g$, $h_0$, $h_1$, and $t$, the bounded recursion on notation $f = BRN(g,h_0,h_1,t)$ is given by:

$$\begin{aligned} f(\varepsilon,\vec{x}) &= g(\vec{x}) \\ f(yi,\vec{x}) &= h_i(y,\vec{x},f(y,\vec{x}))|_{t(y,\vec{x})}, \quad i \in \{0,1\} \end{aligned}$$

  - Bounded primitive recursion: Given $g$, $h$, and $t$, the bounded primitive recursion $f = BPR(g,h,t)$ is given by

$$\begin{aligned} f(\varepsilon,\vec{x}) &= g(\vec{x}) \\ f(y',\vec{x}) &= h(y,\vec{x},f(y,\vec{x}))|_{t(y,\vec{x})} \end{aligned}$$

**Proposition 8.**
- $[\mathscr{I}; C, BRN] = \text{FPtime}$,
- $[\mathscr{I}; C, BRN, BPR] = \text{FPspace}$.

These are well-known results, essentially due to Cobham [Cob65] and Thompson [Tho71], here formulated over $\mathbb{W}$. See [Oit97] or [Oit01] for a reference.

$PR$ is the usual operator for primitive recursion, i.e., $f = PR(g,h)$ means that $f$ is defined by primitive recursion, with $g$ as base function and $h$ as step function.

**Definition 9.** *Given $g,h,t$, the* monotone bounded primitive recursion scheme *is defined by*

$$MBPR(g,h,t) = PR(g,(h|_t)^m).$$

**Remark 10.** *Given a function $t(y,\vec{x})$ in $[\mathscr{I}; C, BRN, MBPR]$, we may define within the same class a function $t^+$, which is non-decreasing in the first argument, i.e., for $y_1 \leq y_2$ we have $|t^+(y_1,\vec{x})| \leq |t^+(y_2,\vec{x})|$, such that for all $y,\vec{x}$, $t(y,\vec{x}) \leq t^+(y,\vec{x})$. For instance:*

$$\begin{aligned} t^+(\varepsilon,\vec{x}) &:= t(\varepsilon,\vec{x}), \\ t^+(y',\vec{x}) &:= \begin{cases} t(y',\vec{x}) & \text{if } |t(y,\vec{x})| \leq |t(y',\vec{x})|, \\ t(y,\vec{x}) & \text{otherwise}. \end{cases} \end{aligned}$$

*In fact, if $t$ is itself non-decreasing in the first argument, then $t^+$ is equal to $t$.*

*Now, we get that*

$$MBPR(g,h,t) = PR(g,(h|_t)^m) = BPR(g,(h|_t)^m,t^+).$$

**Remark 11.** *1. If $h,t \in [\mathscr{I}; C, BRN, MBPR]$ (or $[\mathscr{B}; PC, PRN, MPPR]$), then we have also $h|_t \in [\mathscr{I}; C, BRN, MBPR]$ (or $[\mathscr{B}; PC, PRN, MPPR]$, respectively).*

*2. If $h \in [\mathscr{I}; C, BRN, MBPR]$ (or $[\mathscr{B}; PC, PRN, MPPR]$), then we have $h^m \in [\mathscr{I}; C, BRN, MBPR]$ (or $[\mathscr{B}; PC, PRN, MPPR]$, respectively).*

*Moreover, the function definitions of $h|_t$ and $h^m$ do not make any extra use of the MBPR (or MPPR respectively) scheme (relatively to the definitions of h and t).*

Define by bounded recursion on notation $P(\varepsilon) = \varepsilon$ and $P(xi) = x|_x$ and $D(\varepsilon, x) = x$ and $D(yi, x) = P(x)|_x$. Then $x|_y = D(D(y,x),x)$. This justifies item (i) of the remark above. Item (2) is an obvious consequence of $\preceq$ being decidable in P. The case of $[\mathscr{B}; PC, PRN, MPPR]$ is similar.

**Theorem 12.** $[\mathscr{I}; C, BRN, MBPR] = \mathsf{FPH}$.

*Proof.* We prove that

1. for all $f \in [\mathscr{I}; C, BRN, MBPR]$ there exists a $F \in [\mathscr{B}; PC, PRN, MPPR]$ such that $\forall \vec{x}.\, f(\vec{x}) = F(\vec{x};)$;

2. for all $F \in [\mathscr{B}; PC, PRN, MPPR]$ there exists a $f \in [\mathscr{I}; C, BRN, MBPR]$ such that $\forall \vec{x}, \vec{y}. F(\vec{x}; \vec{y}) = f(\vec{x}, \vec{y})$.

This shows that $[\mathscr{I}; C, BRN, MBPR]$ and $[\mathscr{B}; PC, PRN, MPPR]$ can be identified. Thus, the present statement is a consequence of Proposition 6.

(1) is proven by induction on the complexity of the function definitions. The proof is analogous to the proof of Theorem 3.2 in [Oit97, p. 121]. It uses remark 11.

The proof of (2) is straightforward, by induction on the complexity of the function definition of $F \in [\mathscr{B}; PC, PRN, MPPR]$. It uses remark 7(2). Obviously, the $\mathscr{B}$ functions (4)–(6) are defined using bounded recursion on notation. □

# 3 The theory APH

The applicative theory APH is based on the basic theory B of operations and words, as introduced by Strahm in [Str03, § 3.1], with slight modifications indicated below. In particular, our application is total, while Strahm works in a partial setting.

We formulate B in a standard first order language, with *individual variables* $x, y, z, \ldots$, *individual constants*: $\mathsf{k}, \mathsf{s}$ (combinators); $\mathsf{p}, \mathsf{p}_0, \mathsf{p}_1$ (pairing and projection); $\mathsf{c}_\mathsf{W}$ (case distinction); $\varepsilon$ (empty word); $\mathsf{s}_0, \mathsf{s}_1$ (binary successors), $\mathsf{p}_\mathsf{W}$ (binary predecessor); $\mathsf{s}_\ell, \mathsf{p}_\ell$ (lexicographic successor and predecessor); $\mathsf{c}_\subseteq$ (initial subword relation); $*, \times$ (word concatenation and word multiplication). There is one binary function symbol $\cdot$ for term application, which, however, is usually written by juxtaposition. We have only one unary relation symbol W (binary words), and one binary relation symbol $=$ (equality). *Terms* $(r, s, t, \ldots)$ are build from variables and constants by term application.

We use the usual abbreviations of the framework of applicative theories, which include, in particular,

the following ones:

$$\begin{aligned} 0 &:= \mathsf{s}_0\,\varepsilon,\\ 1 &:= \mathsf{s}_1\,\varepsilon,\\ s \subseteq t &:= \mathsf{c}_{\subseteq}\,st = 0,\\ s \le t &:= \mathsf{l}_{\mathsf{W}}\,s \subseteq \mathsf{l}_{\mathsf{W}}\,t,\\ s * t &:= *st,\\ s \times t &:= \times st. \end{aligned}$$

As we will define $\mathsf{l}_{\mathsf{W}}\,t$ by $1 \times t$, $s \le t$ stands actually for $1 \times s \subseteq 1 \times t$.[2] For $w \in \mathbb{W}$, $\overline{w}$ is the corresponding applicative term.

*Formulas* are usual first-order formulas, build from the atomic formulas $\mathsf{W}(t)$ and $t = s$ by use of negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\to$), and universal ($\forall x$) and existential ($\exists x$) quantification. As abbreviation we use

$$\begin{aligned} \forall x \in \mathsf{W}.\phi &:= \forall x.\mathsf{W}(x) \to \phi,\\ \exists x \in \mathsf{W}.\phi &:= \exists x.\mathsf{W}(x) \wedge \phi,\\ \exists x \le t.\phi &:= \exists x \in \mathsf{W}.x \le t \wedge \phi,\\ t : \mathsf{W} \to \mathsf{W} &:= \forall x \in \mathsf{W}.\mathsf{W}(tx),\\ t : \mathsf{W}^2 \to \mathsf{W} &:= \forall x \in \mathsf{W}.\forall y \in \mathsf{W}.\mathsf{W}(txy). \end{aligned}$$

Note that Strahm formulates $\mathsf{B}$ within the *logic of partial terms*, which includes an extra existence predicate. However, for the present purpose, partiality is not essential and hence we stick to total application. Thus, our logic is standard, *classical* first order logic. For more background on applicative theories see, for instance, [Bee85], [JKS99], or [Kah07].

The non-logical axioms of $\mathsf{B}$ are the following ones:[3]

I. Combinatory algebra and pairing
   (1) $\mathsf{k}xy = x$,
   (2) $\mathsf{s}xyz = xz(yz)$,
   (3) $\mathsf{p}_0(\mathsf{p}xy) = x \wedge \mathsf{p}_1(\mathsf{p}xy) = y$.

II. Definition by cases on $\mathsf{W}$.[4]
   (4) $\mathsf{c}_{\mathsf{W}}\,\varepsilon\,s\,r\,u = s$,
   (5) $\mathsf{W}(t) \to \mathsf{c}_{\mathsf{W}}\,(\mathsf{s}_0\,t)\,s\,r\,u = r$,
   (6) $\mathsf{W}(t) \to \mathsf{c}_{\mathsf{W}}\,(\mathsf{s}_1\,t)\,s\,r\,u = u$,

III. Closure, binary successors, and predecessors
   (7) $\mathsf{W}(\varepsilon) \wedge \forall x.\mathsf{W}(x) \to \mathsf{W}(\mathsf{s}_0\,x) \wedge \mathsf{W}(\mathsf{s}_1\,x)$,

[2] Note that, in $\mathsf{APH}$ the relation $\le$ compares the lengths of the terms, while we used the same symbol before, outside $\mathsf{APH}$, to compare the terms themselves.

[3] In [Str03], Strahm axiomatizes also the tally length of binary words, $\mathsf{l}_{\mathsf{W}}$, since his theory $\mathsf{B}$ does not include word concatenation and word multiplication from the very beginning. In the presence of word multiplication the tally length can be defined by letting $\mathsf{l}_{\mathsf{W}}\,t = 1 \times t$.

[4] Our case distinction checks the last bit of a word, while Strahm uses a case distinction which compares words as a whole.

(8) $\mathsf{s}_0 x \neq \mathsf{s}_1 x \wedge \mathsf{s}_0 x \neq \varepsilon \wedge \mathsf{s}_1 x \neq \varepsilon$,
(9) $\mathsf{p}_\mathsf{W} : \mathsf{W} \to \mathsf{W} \wedge \mathsf{p}_\mathsf{W} \varepsilon = \varepsilon$,
(10) $\mathsf{W}(x) \to \mathsf{p}_\mathsf{W}(\mathsf{s}_0 x) = x \wedge \mathsf{p}_\mathsf{W}(\mathsf{s}_1 x) = x$,
(11) $\mathsf{W}(x) \wedge x \neq \varepsilon \to \mathsf{s}_0(\mathsf{p}_\mathsf{W} x) = x \vee \mathsf{s}_1(\mathsf{p}_\mathsf{W} x) = x$.

IV. Lexicographic successor and predecessor

(12) $\mathsf{s}_\ell : \mathsf{W} \to \mathsf{W} \wedge \mathsf{s}_\ell \varepsilon = 0$,
(13) $\mathsf{W}(x) \to \mathsf{s}_\ell(\mathsf{s}_0 x) = \mathsf{s}_1 x \wedge \mathsf{s}_\ell(\mathsf{s}_1 x) = \mathsf{s}_0(\mathsf{s}_\ell x)$,
(14) $\mathsf{p}_\ell : \mathsf{W} \to \mathsf{W} \wedge \mathsf{s}_\ell \varepsilon = \varepsilon$,
(15) $\mathsf{W}(x) \to \mathsf{p}_\ell(\mathsf{s}_\ell x) = x$,
(16) $\mathsf{W}(x) \wedge x \neq \varepsilon \to \mathsf{s}_\ell(\mathsf{p}_\ell x) = x$.

V. Initial subword relation

(17) $\mathsf{W}(x) \wedge \mathsf{W}(y) \to \mathsf{c}_\subseteq xy = 0 \vee \mathsf{c}_\subseteq xy = 1$,
(18) $\mathsf{W}(x) \to (x \subseteq \varepsilon \leftrightarrow x = \varepsilon)$,
(19) $\mathsf{W}(x) \wedge \mathsf{W}(y) \wedge y \neq \varepsilon \to (x \subseteq y \leftrightarrow x \subseteq \mathsf{p}_\mathsf{W} y \vee x = y)$,
(20) $\mathsf{W}(x) \wedge \mathsf{W}(y) \wedge \mathsf{W}(z) \wedge x \subseteq y \wedge y \subseteq z \to x \subseteq z$.

VI. Word concatenation

(21) $* : \mathsf{W}^2 \to \mathsf{W}$,
(22) $\mathsf{W}(x) \to x * \varepsilon = x$,
(23) $\mathsf{W}(x) \wedge \mathsf{W}(y) \to x * (\mathsf{s}_0 y) = \mathsf{s}_0(x * y) \wedge x * (\mathsf{s}_1 y) = \mathsf{s}_1(x * y)$.

VII. Word multiplication

(24) $\times : \mathsf{W}^2 \to \mathsf{W}$,
(25) $\mathsf{W}(x) \to x \times \varepsilon = \varepsilon$,
(26) $\mathsf{W}(x) \wedge \mathsf{W}(y) \to x \times \mathsf{s}_0 y = (x \times y) * x \wedge x \times \mathsf{s}_1 y = (x \times y) * x$.

**Induction on notation.**

$$f : \mathsf{W} \to \mathsf{W} \wedge \phi(\varepsilon) \wedge (\forall x \in \mathsf{W}.\phi(x) \to \phi(\mathsf{s}_0 x) \wedge \phi(\mathsf{s}_1 x)) \to \forall x \in \mathsf{W}.\phi(x),$$

where $\phi(x)$ is of the form $\exists y \leq f x.\psi(f, x, y)$ for $\psi(f, x, y)$ a *positive and W-free* formula.[5]

This induction is called $(\Sigma^\mathbf{b}_\mathsf{W}\text{-}\mathsf{I}_\mathsf{W})$ in [Str03].

**Monotonicity relation.** It is easy to observe that the monotonicity relation $\preceq$ is polytime decidable. As the theory $\mathsf{B} + (\Sigma^\mathbf{b}_\mathsf{W}\text{-}\mathsf{I}_\mathsf{W})$ allow to represent all polytime functions (as provably total functions in the sense of Definition 14 below), we know that there is term $t_{\chi_\preceq}$ with

1. $\mathsf{B} + (\Sigma^\mathbf{b}_\mathsf{W}\text{-}\mathsf{I}_\mathsf{W}) \vdash t_{\chi_\preceq}\, \overline{w_1}\, \overline{w_2} = \overline{\chi_\preceq(w_1, w_2)}$, for all $w_1, w_2 \in \mathbb{W}$, and
2. $\mathsf{B} + (\Sigma^\mathbf{b}_\mathsf{W}\text{-}\mathsf{I}_\mathsf{W}) \vdash \forall x, y.\mathsf{W}(x) \wedge \mathsf{W}(y) \to t_{\chi_\preceq} xy = 0 \vee t_{\chi_\preceq} xy = 1$.

In the following, we will use $\mathsf{c}_\succeq$ as abbreviation for $\lambda x, y.t_{\chi_\preceq} yx$. Moreover, $s \succeq t$ is used as abbreviation of $\mathsf{c}_\succeq st = 0$. We also introduce quantifier $\exists x \succeq t.\phi$ as abbreviation for $\exists x.\mathsf{W}(x) \wedge x \succeq t \wedge \phi$.

Note that 2. above means that $\mathsf{c}_\succeq$ is total as function from $\mathsf{W}^2 \to \mathsf{W}$. But, of course, $\mathsf{c}_\succeq$ is not total as a binary relation, as we have, for instance, $01 \not\succeq 10$ and $10 \not\succeq 01$.

[5] Positive formulas are defined, as usual, as negation and implication free formulas.

**Remark 13.** *For $u$ and $v$ in $W$, we can show in* APH*:*

1. $u \leq v \to u \preceq 1 \times v$,
2. $u \preceq v \to u \leq v$.

*And we can define a low-level pairing function $\langle \cdot, \cdot \rangle$ and projections $(\cdot)_0$ and $(\cdot)_1$ on $W$, which are, at most, in* FPTIME*, such that* APH *proves for the representing terms:*

3. $u \preceq \langle u, v \rangle$ *and* $v \preceq \langle u, v \rangle$,
4. $(u)_0 \preceq u$ *and* $(u)_1 \preceq u$.

**Monotone induction ($\Sigma^{\mathbf{b}}_{\mathsf{W}}$-MPI).**

$$t : \mathsf{W} \to \mathsf{W} \wedge (\exists x \in \mathsf{W}.\phi(\varepsilon, x)) \wedge (\forall y \in \mathsf{W}.\forall x \in \mathsf{W}.\phi(y,x) \to \exists z \succeq x.\phi(\mathsf{s}_\ell y, z)) \to \forall y \in \mathsf{W}.\exists x \in \mathsf{W}.\phi(y,x),$$

where $\phi(y,x)$ is of the form $x \leq t\,y \wedge \psi(t,y,x)$ for $\psi(t,y,x)$ a *positive and W-free* formula *not containing disjunctions*. For the reason of the exclusion of disjunctions, see remark 20 below.

Essentially, APH is equal to Strahm's theory PT plus the monotone induction scheme ($\Sigma^{\mathbf{b}}_{\mathsf{W}}$-MPI).

# 4 The lower bound

**Definition 14.** *A function $F : \mathbb{W}^n \to \mathbb{W}$ is called* provably total in APH*, if there exists a closed term $t_F$ such that*

1. $\mathsf{APH} \vdash t_F\,\overline{w_1} \ldots \overline{w_n} = \overline{F(w_1, \ldots, w_n)}$ *for all $w_1, \ldots, w_n \in \mathbb{W}$, and*
2. $\mathsf{APH} \vdash t_F : W^n \to W$.

Using the result of [Str03, § 4] about the provably total function in Strahm's theory corresponding to FPTIME, it remains to show that functions defined by the monotone bounded primitive recursion scheme $MBPR(g,h,t)$ are provably total in APH.

So, let us assume that $g$, $h$, and $t$ are provably total in APH, and $f$ be defined as $MBPR(g,h,t) = PR(g,(h|_t)^m)$.

Now, in APH, let

$$f(\varepsilon, \vec{z}) = g(\vec{z})$$
$$f(\mathsf{s}_\ell y, \vec{z}) = \begin{cases} h|_t(y, \vec{z}, f(y,\vec{z})) & \text{if } f(y,\vec{z}) \preceq h|_t(y,\vec{z},f(y,\vec{z})) \\ f(y,\vec{z}) & \text{otherwise} \end{cases}$$

and we show by monotone induction that $\forall y \in \mathsf{W}.\exists x \in \mathsf{W}.x \leq t_f(y,\vec{z}) \wedge f(y,\vec{z}) = x$, where

$$t_f(y,\vec{x}) = \begin{cases} g(\vec{z}) & \text{if } y = \varepsilon, \\ t^+(y,\vec{z}) & \text{otherwise.} \end{cases}$$

*Induction base:* As $f(\varepsilon,\vec{z}) = g(\vec{z})$, and $g$ is provably total in APH, we have $\exists x \in \mathsf{W}.x \leq g(\vec{z}) \wedge f(\varepsilon,\vec{z}) = x$.

*Induction step:* We have to show that $\forall y \in \mathsf{W}.\forall x \in \mathsf{W}.x \leq t_f(y,\vec{z}) \wedge f(y,\vec{z}) = x \to \exists x_1 \succeq x.\ x_1 \leq t_f(\mathsf{s}_\ell y,\vec{z}) \wedge f(\mathsf{s}_\ell y,\vec{z}) = x_1$.

By definition,

$$f(\mathsf{s}_\ell y,\vec{z}) = \begin{cases} h|_t(y,\vec{z},f(y,\vec{z})) & \text{if } f(y,\vec{z}) \preceq h|_t(y,\vec{z},f(y,\vec{z})), \\ f(y,\vec{z}) & \text{otherwise.} \end{cases}$$

In the first case, the assertion follows immediately from the condition $f(y,\vec{z}) \preceq h|_t(y,\vec{z},f(y,\vec{z}))$.
In the second case, the assertion follows immediately from the premise (choosing $x_1 := x$).
Thus, we can conclude by monotone induction that $\forall y \in \mathsf{W}.\exists x \in \mathsf{W}.x \leq t_f(y,\vec{z}) \wedge f(\mathsf{s}_\ell y,\vec{z}) = x$.
Thus, we get the following result:

**Lemma 15.** *The provably total functions of* APH *include* FPH.

# 5 The upper bound

The proof of the upper bound follows quite closely the proof of the upper bound of Strahm for his theory PT in [Str03, § 6]. For it, one reformulates the theory first in Gentzen's classical sequence calculus, and proves partial cut elimination, such that the remaining cuts are restricted to positive formulas. In a second step, one realizes positive derivations with realizers from the appropriate complexity class. In this step, one uses the open term model $\mathscr{M}(\lambda\eta)$ of the applicative ground structure, which is based on the usual $\lambda\eta$ reduction of the untyped $\lambda$-calculus. In fact, $\eta$ allows us to treat extensionality of operations, i.e., we may add the following axiom to APH:

(Ext) $\forall f,g.(\forall x.fx = gx) \to f = g$.

For the treatment of APH, we will follow Strahm's proof for PT, and check only, how to take care of our additional monotone induction scheme ($\mathbf{\Sigma}^{\mathbf{b}}_{\mathsf{W}}$-MPI).

Let $\mathsf{APH}^+$ the Gentzen-style sequent calculus reformulation of APH such that all main formulas of non-logical axioms and rules are positive. In this calculus, the monotone induction ($\mathbf{\Sigma}^{\mathbf{b}}_{\mathsf{W}}$-MPI) is rewritten as the following rule:

$$\frac{\begin{array}{c}\Gamma,\mathsf{W}(u) \Rightarrow \mathsf{W}(tu),\Delta \\ \Gamma \Rightarrow \exists n.\mathsf{W}(n) \wedge \phi(\varepsilon,n),\Delta \\ \Gamma,\mathsf{W}(a),\mathsf{W}(b),\phi(a,b) \Rightarrow \exists m \succeq b.\phi(\mathsf{s}_\ell a,m),\Delta\end{array}}{\Gamma,\mathsf{W}(s) \Rightarrow \exists n.\mathsf{W}(n) \wedge \phi(s,n),\Delta},$$

where $\phi(s,n)$ is of the form $n \leq ts \wedge \psi(t,s,n)$ for $\psi(t,s,n)$ a positive and W-free formula which does not contain disjunctions.

We write $\mathsf{APH}^+ \vdash \Gamma \Rightarrow \Delta$ if the sequent $\Gamma \Rightarrow \Delta$ is derivable in $\mathsf{APH}^+$, and $\mathsf{APH}^+ \vdash_* \Gamma \Rightarrow \Delta$ if it has a proof where all cut formulas are *positive*.

## 5.1 Partial cut elimination

**Theorem 16** (Partial cut elimination, cf. [Str03, Theorem 12])**.** *For all sequents* $\Gamma \Rightarrow \Delta$, $\mathsf{APH}^+ \vdash \Gamma \Rightarrow \Delta$ *implies* $\mathsf{APH}^+ \vdash_* \Gamma \Rightarrow \Delta$.

We only have to check that the main formulas of our induction rules are positive, but that is the case since, in particular, $\exists m \succeq b.\phi(\mathsf{s}_\ell a,m)$ is positive.

**Corollary 17** (cf. [Str03, Corollary 13])**.** *If* $\Gamma \Rightarrow \Delta$ *is a sequent of positive formulas with* $\mathsf{APH}^+ \vdash \Gamma \Rightarrow \Delta$, *then there is a* $\mathsf{APH}^+$ *derivation of* $\Gamma \Rightarrow \Delta$ *which contains only positive formulas.*

## 5.2 Realizability

**Definition 18.** *Let $\rho \in \mathbb{W}$ and $\phi$ a positive formula. Then $\rho \triangleright \phi$ is inductively defined as follows:*[6]

$$\begin{array}{rll}
\rho \triangleright W(t) & \text{if} & \mathscr{M}(\lambda\eta) \models t = \overline{\rho}, \\
\rho \triangleright (t_1 = t_2) & \text{if} & \rho = \varepsilon \text{ and } \mathscr{M}(\lambda\eta) \models t_1 = t_2, \\
\rho \triangleright (\phi \wedge \psi) & \text{if} & \rho = \langle \rho_0, \rho_1 \rangle \text{ and } \rho_0 \triangleright \phi \text{ and } \rho_1 \triangleright \psi, \\
\rho \triangleright (\phi \vee \psi) & \text{if} & \rho = \langle i, \rho_0 \rangle \text{ and either } i = 0 \text{ and } \rho_0 \triangleright \phi \text{ or } i = 1 \text{ and } \rho_0 \triangleright \psi, \\
\rho \triangleright (\forall x.\phi(x)) & \text{if} & \rho \triangleright \phi(u) \text{ for a fresh variable } u, \\
\rho \triangleright (\exists x.\phi(x)) & \text{if} & \rho \triangleright \phi(t) \text{ for some term } t.
\end{array}$$

$\rho$ realizes a sequence $\Delta$ of $n$ formulas $\phi_1, \ldots, \phi_n$, if $\rho = \langle i_2, \rho_0 \rangle$, $1 \leq i \leq n$, $i_2$ the dyadic representation of the natural number $i$, and $\rho_0 \triangleright \phi_i$.

To improve readability, we use the following abbreviations regarding our low-level pairing in the context of realizability: When we $\rho$ realizes a conjunction $\phi \wedge \psi$, $\mathit{left}(\rho)$ for the $(\rho)_0$, i.e., the realizer of $\phi$, and, analogously $\mathit{right}(\rho)$ for the realizer $(\rho)_1$ of $\psi$. When $\rho$ realizes a sequence $\phi_1, \ldots, \phi_n$, we write $\mathit{no}(\rho)$ for $(\rho)_0$, i.e., the index of the realized formula, and $\mathit{sel}(\rho)$ for $(\rho)_1$, the realizer of the selected formula.

**Theorem 19** (Realizability for $\mathsf{APH}^+$, cf. [Str03, Theorem 15])**.** *Let $\Gamma \Rightarrow \Delta$ be a sequent of positive formulas with $\Gamma = \phi_1, \ldots, \phi_n$ and assume that $\mathsf{APH}^+ \vdash_* \Gamma[\vec{u}] \Rightarrow \Delta[\vec{u}]$. Then there exists a function $F : \mathbb{W}^n \to \mathbb{W}$ in $\mathsf{FPH}$ such that for all terms $\vec{s}$ and all $\rho_1, \ldots, \rho_n \in \mathbb{W}$:*

$$\rho_1 \triangleright \phi_1[\vec{s}], \ldots, \rho_n \triangleright \phi_n[\vec{s}] \qquad \Longrightarrow \qquad F(\rho_1, \ldots, \rho_n) \triangleright \Delta[\vec{s}].$$

The proof runs by induction on the length of a quasi cut-free derivation. We have only to check the case of our monotone induction rule, as all other cases are like in [Str03].

By induction hypothesis, we get for the three premises:

$$\Gamma, \mathsf{W}(u) \Rightarrow \mathsf{W}(t\,u), \Delta \tag{1}$$

$$\Gamma \Rightarrow \exists n.\mathsf{W}(n) \wedge \phi(\varepsilon, n), \Delta \tag{2}$$

$$\Gamma, \mathsf{W}(a), \mathsf{W}(b), \phi(a,b) \Rightarrow \exists m \succeq b.\phi(\mathsf{s}_\ell\, a, m), \Delta \tag{3}$$

that there are functions $T$, $G$ and $H$ in $\mathsf{FPH}$ such that for all $\vec{\rho}, \sigma, \tau, \upsilon$:

$$\begin{array}{rcl}
\vec{\rho} \triangleright \Gamma[\vec{s}] & \Rightarrow & T(\sigma, \vec{\rho}) \triangleright \mathsf{W}(t[\vec{s}](\sigma)), \Delta[\vec{s}] \\
\vec{\rho} \triangleright \Gamma[\vec{s}] & \Rightarrow & G(\vec{\rho}) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\varepsilon, n)[\vec{s}], \Delta[\vec{s}] \qquad (4) \\
\vec{\rho} \triangleright \Gamma[\vec{s}], \upsilon \triangleright \phi(\sigma, \tau)[\vec{s}] & \Rightarrow & \tilde{H}(\sigma, \vec{\rho}, \tau, \upsilon) \triangleright \exists m \succeq \tau.\phi(\mathsf{s}_\ell\, \sigma, m)[\vec{s}], \Delta[\vec{s}] \qquad (5)
\end{array}$$

Now, we need a function $F$ in $\mathsf{FPH}$, such that

$$\vec{\rho} \triangleright \Gamma[\vec{s}] \quad \Rightarrow \quad F(\sigma, \vec{\rho}) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\sigma, n)[\vec{s}], \Delta[\vec{s}] \tag{6}$$

We set

$$H(\sigma, \vec{\rho}, \omega) = \langle 1, \langle \mathit{left}(\mathit{sel}(\tilde{H}(\sigma, \vec{\rho}, \mathit{left}(\omega), \mathit{right}(\omega)))), \\ \mathit{right}(\mathit{right}(\mathit{sel}(\tilde{H}(\sigma, \vec{\rho}, \mathit{left}(\omega), \mathit{right}(\omega)))))\rangle\rangle.$$

[6] Here $\langle \cdot, \cdot \rangle$ is a low-level pairing function on binary words, with its projections $(\cdot)_0$ and $(\cdot)_1$.

This definition looks quite involved, its idea is, however, straightforward: when, according to (5), $\tilde{H}$ will realize a formula of the form $\exists m \succeq \tau.\phi(\mathsf{s}_\ell\,\sigma, m)[\vec{s}]$, $H$ is supposed to realize $\exists m.\mathsf{W}(m) \wedge \phi(\mathsf{s}_\ell\,\sigma, m)[\vec{s}]$. Thus we have to "cut out" the second conjunct $m \succeq \tau$ under the existential quantifier ($\mathsf{W}(m)$ is the first conjunct which is not visible in the abbreviation $\exists m \succeq \tau$).

Before defining the function $F$ which should realize the conclusion of our rule, we define an auxiliary function $F'$ which returns a pair, having the intended value of $F$ as its second component. The first component serves only to guarantee the monotonicity.

So, $F'(\sigma, \vec{\rho}, \tau)$ is defined by monotone recursion as:

$$F'(\varepsilon, \vec{\rho}) = \langle \varepsilon, G(\vec{\rho})\rangle,$$

$$F'(\mathsf{s}_\ell\,\sigma, \vec{\rho}) = \begin{cases} F'(\sigma, \vec{\rho}) & \text{if } no(right(F'(\sigma, \vec{\rho}))) \neq 1 \\ & \quad (F \text{ will realize one of the } \Delta\text{s}), \\ \langle F'(\sigma, \vec{\rho}), T(\sigma, \vec{\rho})\rangle & \text{if } no(right(F'(\sigma, \vec{\rho}))) = 1 \text{ and } no(T(\sigma, \vec{\rho})) \neq 1 \\ & \quad (T \text{ realizes one of the } \Delta\text{s}), \\ \langle \varepsilon, H(\sigma, \vec{\rho}, sel(right(F'(\sigma, \vec{\rho}))))\rangle & \text{otherwise.} \end{cases}$$

With this function, $F(\sigma, \vec{\rho})$ is defined as $right(F'(\sigma, \vec{\rho}))$.

To check (6) we can use a straightforward (meta-)induction on $\sigma$:

$\sigma = \varepsilon$: Given $\vec{\rho} \triangleright \Gamma[\vec{s}]$, in this case, $F(\varepsilon, \vec{\rho}) = G(\vec{\rho}) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\varepsilon, n)[\vec{s}], \Delta[\vec{s}]$ by (4).

$\mathsf{s}_\ell\,\sigma$: In the first and second case, we know that one of the side formulas $\Delta[\vec{s}]$ is realized, and, of course, $F(\mathsf{s}_\ell\,\sigma, \vec{\rho})$ realizes one of these side formulas, too. In the third case, we have to show that

$$H(\sigma, \vec{\rho}, sel(F(\sigma, \vec{\rho}))) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\mathsf{s}_\ell\,\sigma, n)[\vec{s}], \Delta[\vec{s}].$$

We know that $no(F(\sigma, \vec{\rho})) = 1$, thus, using the induction hypothesis, we know that the first formula of the sequence is realized, i.e.,

$$sel(F(\sigma, \vec{\rho})) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\sigma, n)[\vec{s}].$$

That means, $left(sel(F(\sigma), \vec{\rho})) = \tau$ for a $\tau$ with $right(sel(F(\sigma), \vec{\rho})) \triangleright \phi(\sigma, \tau)[\vec{s}]$. By definition of $H(\sigma, \vec{\rho}, sel(F(\sigma, \vec{\rho})))$ is $\tilde{H}(\sigma, \vec{\rho}, left(sel(F(\sigma, \vec{\rho}))), right(sel(F(\sigma, \vec{\rho}))))$. Letting $\tau$ be as above the term $left(sel(F(\sigma), \vec{\rho}))$, and $\upsilon := right(sel(F(\sigma, \vec{\rho})))$, we get from (5) that

$$H(\sigma, \vec{\rho}, sel(F(\sigma, \vec{\rho})))m = \\ \tilde{H}(\sigma, \vec{\rho}, left(sel(F(\sigma, \vec{\rho}))), right(sel(F(\sigma, \vec{\rho})))) \triangleright \exists m \succeq left(sel(F(\sigma, \vec{\rho}))).\phi(\mathsf{s}_\ell\,\sigma, m)[\vec{s}], \Delta[\vec{s}].$$

The remaining coding serves to get rid of the redundant monotonicity condition.

It remains to show that $F$ is in FPH. For it, we only need to check that the step function $F'$ is of the form $h|_t$, with $h$ and $t$ in $[\mathscr{I}; C, BRN, MBPR]$, and monotone.

That the step function is bounded follows essentially as in the proof of [Str03, Theorem 15] with the fact that the formula $\phi(y, n)$ has the shape $n \leq t\,y \wedge \psi(t, y, n)$.

Monotonicity: as in the first and second case, the function stays constant, we only have to check that the value is greater or equal (in the sense of our monotonicity relation $\preceq$) as the recursive argument $F'(\sigma, \vec{\rho})$. This is trivial in the first case (where it is equal), and follows in the second case from the fact that $F'(\sigma, \vec{\rho})$ is coded in the first argument of the pair. In the third case, we have to show that, for all $\sigma$, $F'(\sigma, \vec{\rho}) \preceq \langle \varepsilon, H(\sigma, \vec{\rho}, sel(right(F'(\sigma, \vec{\rho}))))\rangle$. From the case distinction, we know, that $right(F'(\sigma, \vec{\rho})) =$

$\langle 1, sel(right(F'(\sigma,\vec{\rho})))\rangle$, and $sel(right(F'(\sigma,\vec{\rho}))) \triangleright \exists n.\mathsf{W}(n) \wedge \phi(\sigma,n)[\vec{s}]$, i.e., $sel(right(F'(\sigma,\vec{\rho})))$ is of the form $\langle \omega_0, \omega_1\rangle$ with $\omega_1 \triangleright \phi(\sigma,\omega_0)[\vec{s}]$. On the other hand,

$$\begin{aligned}
&H(\sigma,\vec{\rho},sel(right(F'(\sigma,\vec{\rho}))))\\
&= \langle 1, \langle left(sel(\tilde{H}(\sigma,\vec{\rho},left(sel(right(F'(\sigma,\vec{\rho})))),right(right(sel(right(F'(\sigma,\vec{\rho}))))))),\\
&\qquad right(right(sel(\tilde{H}(\sigma,\vec{\rho},left(sel(right(F'(\sigma,\vec{\rho})))),right(sel(right(F'(\sigma,\vec{\rho}))))))))\rangle\rangle\\
&= \langle 1, \langle left(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1))), right(right(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1))))\rangle\rangle.
\end{aligned}$$

According to (5) and the condition of the case distinction we have

$$sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1)) \triangleright \exists m \succeq \omega_0.\phi(\mathsf{s}_\ell\,\sigma,m)[\vec{s}]$$

or, more detailed,

$$sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1)) \triangleright \exists m.\mathsf{W}(m) \wedge m \succeq \omega_0 \wedge \phi(\mathsf{s}_\ell\,\sigma,m)[\vec{s}].$$

From the second conjunct we can conclude, $\omega_0 \preceq left(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1)))$. It remains to show that $\omega_1 \preceq right(right(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1))))$. We have $\omega_1 \triangleright \phi(\sigma,\omega_0)[\vec{s}]$ and $right(right(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1)))) \triangleright \phi(\mathsf{s}_\ell\,\sigma,left(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1))))[\vec{s}]$. Now, it is important that $\phi$ is a positive, W-free formula *without disjunction*. For these class of formulas, the realizers do not depend on the terms occurring in them (as long as they are realizable, of course). Thus, $\omega_1$ and $right(right(sel(\tilde{H}(\sigma,\vec{\rho},\omega_0,\omega_1))))$ are equal. Now, the monotonicity follows from the properties we have for the monotonicity relation together with the pairing (see Remark 13).

**Remark 20.** *The proof of the monotonicity property of the step function depends on our restriction to disjunction-free formulas in the monotone induction scheme. In fact, if we allow disjunctions, the monotonicity is not any longer guaranteed, as, depending on the terms, different disjuncts could be realized and the value of the realizers may differ. In fact, disjunction has a "non-monotonic" flavor. However, it is not clear whether one can make any use of disjunction to enlarge the class of provably total functions. So, we pose as a question:*

**Question 21.** *What is the class of provably total functions of* APH *if the monotone induction scheme allows disjunctions in the formula $\phi(y,n)$?*

The final result follows now as a corollary:

**Corollary 22** (cf. [Str03, Corollary 16])**.** *Let $t$ be a closed term and assume that*

$$\mathsf{APH}^+ \vdash W(u_1) \wedge \cdots \wedge W(u_n) \Rightarrow W(t\,u_1 \ldots u_n),$$

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

$$\mathscr{M}(\lambda\,\eta) \models t\,\overline{w_1}\ldots\overline{w_n} = \overline{F(w_1,\ldots,w_n)}.$$

# References


[BALO1x] Amir M. Ben-Amram, Bruno Loff, and Isabel Oitavem. Monotonicity constraints in characterizations of pspace. *Journal of Logic and Computation*, 201x. to appear.

[BC92] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the poly-time functions. *Computational Complexity*, 2:97–110, 1992.

[Bee85] Michael Beeson. *Foundations of Constructive Mathematics*. Ergebnisse der Mathematik und ihrer Grenzgebiete; 3.Folge, Band 6. Springer, 1985.

[Can02] Andrea Cantini. Polytime, combinatory logic and positive safe induction. *Archive for Mathematical Logic*, 41(2):169–189, 2002.

[Clo99] Peter Clote. Computational models and function algebras. In E. Griffor, editor, *Handbook of Computability Theory*, pages 589–681. Elsevier, 1999.

[Cob65] A. Cobham. The intrinsic computational difficulty of functions. In *Logic, Methodology, and Philsophy of Science II*, pages 24–30. North-Holland, 1965.

[Fef75] Solomon Feferman. A language and axioms for explicit mathematics. In J. Crossley, editor, *Algebra and Logic*, volume 450 of *Lecture Notes in Mathematics*, pages 87–139. Springer, 1975.

[Fef79] Solomon Feferman. Constructive theories of functions and classes. In M. Boffa, D. van Dalen, and K. McAloon, editors, *Logic Colloquium 78*, pages 159–224. North–Holland, 1979.

[JKS99] Gerhard Jäger, Reinhard Kahle, and Thomas Strahm. On applicative theories. In A. Cantini, E. Casari, and P. Minari, editors, *Logic and Foundation of Mathematics*, pages 88–92. Kluwer, 1999.

[Kah07] Reinhard Kahle. *The applicative realm*, volume 40 of *Textos de Matemática*. Departamento de Matemática, Universidade de Coimbra, 2007. Habilitationsschrift, Fakultät für Informations- und Kommunikationswissenschaften, Universität Tübingen.

[Oit97] Isabel Oitavem. New recursive characterizations of the elementary functions and the functions computable in polynomial space. *Revista Matemática de la Universidad Complutense de Madrid*, 10(1):109–125, 1997.

[Oit01] Isabel Oitavem. Implicit characterizations of *Pspace*. In R. Kahle, P. Schroeder-Heister, and R. Stärk, editors, *Proof Theory in Computer Science*, volume 2183 of *Lecture Notes in Computer Science*, pages 170–190. Springer, 2001.

[Sch06] Helmut Schwichtenberg. An arithmetic for polynomial-time computation. *Theoretical Computer Science*, 357(1):202–214, 2006.

[Str97] Thomas Strahm. Polynomial time operations in explicit mathematics. *Journal of Symbolic Logic*, 62(2):575–594, 1997.

[Str03] Thomas Strahm. Theories with self-application and computational complexity. *Information and Computation*, 185:263–297, 2003.

[Tho71] D. Thompson. Subrecursion: Machine-independent notions of computability in restricted time and storage. *Mathematical Systems Theory*, 6(1):3–15, 1971.