Weak theories of operations and types

Thomas Strahm

Institut für Informatik und angewandte Mathematik, Universität Bern

Logic Colloquium 2008

General aims of this talk

In this talk we will discuss

- weak systems of operations and types in the spirit of Feferman's explicit mathematics
- uniform proof-theoretic characterizations of various classes of computational complexity in this setting
- relationship to traditional bounded arithmetic
- issues of feasibility in higher types
- some aspects of self-referential truth

Explicit mathematics

Systems of explicit mathematics have been introduced by Feferman in 1975. They have been employed in foundational works in various ways:

- foundations of constructive mathematics
- proof theory of subsystems of second order arithmetic and set theory; foundational reductions
- logical foundations of functional programming languages
- universes and higher reflection principles
- formal proof-theoretic framework for abstract computations from ordinary and generalized recursion theory

Informal applicative setting

Informal applicative setting

- Untyped universe of operations or rules, which can be freely applied to eachother.

Informal applicative setting

- Untyped universe of operations or rules, which can be freely applied to eachother.
- Self-application is meaningful, though not necessarily total.

Informal applicative setting

- Untyped universe of operations or rules, which can be freely applied to eachother.
- 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.

Informal applicative setting

- Untyped universe of operations or rules, which can be freely applied to eachother.
- 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.

Informal applicative setting (ctd.)

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

Informal applicative setting (ctd.)

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

- s_0 and s_1: binary successors on $\mathbb{W}$ with predecessor p_W

Informal applicative setting (ctd.)

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

- 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

Informal applicative setting (ctd.)

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

- 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

Informal applicative setting (ctd.)

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

- 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

The logic of partial terms

The logic of partial terms (LPT) due to Beeson/Feferman is a modification of first-order predicate logic taking into account partial functions.

The logic of partial terms

The logic of partial terms (LPT) due to Beeson/Feferman is a modification of first-order predicate logic taking into account partial functions.

- Variables range over defined objects only

The logic of partial terms

The logic of partial terms (LPT) due to Beeson/Feferman is a modification of first-order predicate logic taking into account partial functions.

- Variables range over defined objects only
- (Composed) terms do not necessarily denote and $t\downarrow$ signifies that t has a value

The logic of partial terms

The logic of partial terms (LPT) due to Beeson/Feferman is a modification of first-order predicate logic taking into account partial functions.

- Variables range over defined objects only
- (Composed) terms do not necessarily denote and $t\downarrow$ signifies that t has a value
- The usual quantifier axioms of predicate logic are modified, e.g. we have

$$A(t) \land t\downarrow \rightarrow (\exists x)A(x)$$

The logic of partial terms

The logic of partial terms (LPT) due to Beeson/Feferman is a modification of first-order predicate logic taking into account partial functions.

- Variables range over defined objects only
- (Composed) terms do not necessarily denote and $t\downarrow$ signifies that t has a value
- The usual quantifier axioms of predicate logic are modified, e.g. we have
$$A(t) \wedge t\downarrow \; \rightarrow \; (\exists x)A(x)$$
- Strictness axioms claim that terms occurring in positive atoms are defined

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, d_W, ϵ, s_0, s_1, p_W, s_ℓ, p_ℓ, $\mathsf{c}_\subseteq$, l_W ...
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, d_W, ϵ, s_0, s_1, p_W, s_ℓ, p_ℓ, $\mathsf{c}_\subseteq$, l_W ...
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \dots t_n \; := \; (\dots(t_1 \circ t_2) \circ \dots \circ t_n)$

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{d}_W, \epsilon, \mathsf{s}_0, \mathsf{s}_1, \mathsf{p}_W, \mathsf{s}_\ell, \mathsf{p}_\ell, \mathsf{c}_\subseteq, \mathsf{l}_W \ldots$
- relation symbols $=, \downarrow, W$
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \; := \; (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \; := \; t_1\!\downarrow \vee\, t_2\!\downarrow \,\rightarrow\, t_1 = t_2$

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{d}_W, \epsilon, \mathsf{s}_0, \mathsf{s}_1, \mathsf{p}_W, \mathsf{s}_\ell, \mathsf{p}_\ell, \mathsf{c}_\subseteq, \mathsf{l}_W \dots$
- relation symbols $=, \downarrow, W$
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \dots t_n \ := \ (\dots(t_1 \circ t_2) \circ \dots \circ t_n)$
- $t_1 \simeq t_2 \ := \ t_1{\downarrow} \vee t_2{\downarrow} \rightarrow t_1 = t_2$
- $t \in W \ := \ W(t)$

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, d_W, ϵ, s_0, s_1, p_W, s_ℓ, p_ℓ, $\mathsf{c}_\subseteq$, l_W ...
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \dots t_n \; := \; (\dots(t_1 \circ t_2) \circ \dots \circ t_n)$
- $t_1 \simeq t_2 \; := \; t_1\!\downarrow \vee\, t_2\!\downarrow \,\rightarrow\, t_1 = t_2$
- $t \in \mathsf{W} \; := \; \mathsf{W}(t)$
- $t : \mathsf{W}^k \rightarrow \mathsf{W} \; := \; (\forall x_1 \dots x_k \in \mathsf{W}) t x_1 \dots x_k \in \mathsf{W}$

The basic applicative language $\mathcal{L}$

$\mathcal{L}$ is a first order language for the logic of partial terms:

- constants $\mathsf{k}, \mathsf{s}, \mathsf{p}, \mathsf{p}_0, \mathsf{p}_1, \mathsf{d}_\mathsf{W}, \epsilon, \mathsf{s}_0, \mathsf{s}_1, \mathsf{p}_\mathsf{W}, \mathsf{s}_\ell, \mathsf{p}_\ell, \mathsf{c}_\subseteq, \mathsf{l}_\mathsf{W} \ldots$
- relation symbols $=, \downarrow, \mathsf{W}$
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \; := \; (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \; := \; t_1\downarrow \vee\, t_2\downarrow \;\rightarrow\; t_1 = t_2$
- $t \in \mathsf{W} \; := \; \mathsf{W}(t)$
- $t : \mathsf{W}^k \rightarrow \mathsf{W} \; := \; (\forall x_1 \ldots x_k \in \mathsf{W}) t x_1 \ldots x_k \in \mathsf{W}$
- $t : \mathsf{W}^\mathsf{W} \times \mathsf{W} \rightarrow \mathsf{W} \; := \; (\forall f \in \mathsf{W} \rightarrow \mathsf{W})(\forall x \in \mathsf{W}) t f x \in \mathsf{W}$

The basic theory of operations and words B

The logic of B is the 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 $\mathsf{c}_\subseteq$, length of words l_W

Consequences of the partial combinatory algebra axioms

As usual in untyped applicative settings we have:

Lemma (Explicit definitions and fixed points)

1. *For each $\mathcal{L}$ term t there exists an $\mathcal{L}$ 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 $\mathcal{L}$ term fix so that*

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

Standard models

Example (Recursion-theoretic model *PRO*)

Take the universe of binary words and interpret application $\circ$ as partial recursive function application in the sense of o.r.t.

Standard models

Example (Recursion-theoretic model *PRO*)

Take the universe of binary words and interpret application $\circ$ as partial recursive function application in the sense of o.r.t.

Example (The open term model $\mathcal{M}(\lambda\eta)$)

- Take the universe of open terms
- Consider the usual reduction of the extensional untyped lambda calculus $\lambda\eta$
- Application is juxtaposition
- Two terms are equal if they have a common reduct
- W denotes those terms that reduce to a "standard" word $\overline{w}$

Natural induction principles

Natural induction principles

Σ^b_W-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Natural induction principles

Σ^b_{W}-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Σ^b_{W} notation induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$

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

Natural induction principles

Σ^b_{W}-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Σ^b_{W} notation induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$

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

Σ^b_{W} lexicographic induction on W, $(\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\ell})$

$$f : \mathsf{W} \to \mathsf{W} \wedge A(\epsilon) \wedge (\forall x \in \mathsf{W})(A(x) \to A(\mathsf{s}_\ell x)) \to (\forall x \in \mathsf{W})A(x)$$

Deriving bounded recursions

Using the fixed point theorem one proves the following lemma:

Bounded recursion on notation

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

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

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

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

Similarly, bounded unary recursion is derivable in $\mathsf{B} + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_\ell)$.

Provably total functions

Definition

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

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

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

Let $\boldsymbol{\tau}(\mathsf{T}) = \{F : F \text{ provably total in } \mathsf{T}\}$.

Four natural applicative systems

The four systems PT, PTLS, PS, LS

$$\mathsf{PT} := \mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}}) \qquad \mathsf{PTLS} := \mathsf{B}(*) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I}_{\mathsf{W}})$$

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

Four natural applicative systems

The four systems PT, PTLS, PS, LS

$$\mathsf{PT} := \mathsf{B}(*, \times) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W}) \qquad \mathsf{PTLS} := \mathsf{B}(*) + (\Sigma^{\mathsf{b}}_{\mathsf{W}}\text{-}\mathsf{I_W})$$

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

Theorem (S '03)

We have the following lower bounds:

1. FPTIME *is contained in* $\boldsymbol{\tau}$(PT),
2. FPTIMELINSPACE *is contained in* $\boldsymbol{\tau}$(PTLS),
3. FPSPACE *is contained in* $\boldsymbol{\tau}$(PS),
4. FLINSPACE *is contained in* $\boldsymbol{\tau}$(LS).

Classical systems of bounded arithmetic and PT

- Ferreira's system PTCA^+ is directly contained in PT
- PTCA^+ corresponds to Buss' S^1_2
- The Melhorn-Cook-Urquhart basic feasible functionals resp. the system PV^ω are directly contained in PT (see later)

Upper bounds: partial cut elimination

Upper bounds: partial cut elimination

- In order to extract computational information from proofs, we need a sequent-style reformulation of our systems and a preparatory partial cut-elimination result

Upper bounds: partial cut elimination

- In order to extract computational information from proofs, we need a sequent-style reformulation of our systems and a preparatory partial cut-elimination result
- In the following we let $\Gamma \Rightarrow \Delta$ range over sequents of formulas

Upper bounds: partial cut elimination

- In order to extract computational information from proofs, we need a sequent-style reformulation of our systems and a preparatory partial cut-elimination result
- In the following we let $\Gamma \Rightarrow \Delta$ range over sequents of formulas
- Since the main formulas in the non-logical axioms and rules are positive, we can reduce all non-positive cuts; $\vdash_{\star}$ denotes provability restricted to positive cuts.

Upper bounds: partial cut elimination

- In order to extract computational information from proofs, we need a sequent-style reformulation of our systems and a preparatory partial cut-elimination result
- In the following we let $\Gamma \Rightarrow \Delta$ range over sequents of formulas
- Since the main formulas in the non-logical axioms and rules are positive, we can reduce all non-positive cuts; $\vdash_{\star}$ denotes provability restricted to positive cuts.
- We establish upper bounds directly for an extension of our systems by the axioms of *totality of application* and *extensionality of operations*.

Upper bounds: realizability

Definition (Realizability for positive formulas)

Let A be a positive formula and $\rho \in \mathbb{W}$.

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

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

Upper bounds: Main Lemma

Lemma (Realizability for PT)

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

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

Similar realizability theorems hold for the systems PTLS, PS, and LS.

The main theorem (concluded)

Theorem (S '03)

We have the following characterizations:

1. τ(PT) *equals* FPTIME,
2. τ(PTLS) *equals* FPTIMELINSPACE,
3. τ(PS) *equals* FPSPACE,
4. τ(LS) *equals* FLINSPACE.

Basic feasible functionals (Melhorn-Cook-Urquhart)

General area of higher type complexity theory.

In particular: feasible functionals of higher type.

Most robust class: basic feasible functionals BFF.

Various kinds of interesting characterizations:

- function algebra, typed lambda calculus (Melhorn, Cook-Urquhart)
- programming languages (Cook-Kapron, Irwin-Kapron-Royer)
- oracle Turing machines (Cook-Kapron, Seth)
- bounded arithmetic (Seth, Ignjatovic)

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words
- a type two functional for bounded recursion on notation

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words
- a type two functional for bounded recursion on notation
- extensionality (optional)

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words
- a type two functional for bounded recursion on notation
- extensionality (optional)
- NP induction

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words
- a type two functional for bounded recursion on notation
- extensionality (optional)
- NP induction

The system PV^ω

PV^ω is a typed formal system whose terms denote exactly the basic feasible functionals. Originally used by Cook and Urquhart for functional and realizability interpretations of intuitionistic systems of bounded arithmetic.

PV^ω includes:

- typed lambda calculus over the base type of binary words (or natural numbers)
- basic operations on words
- a type two functional for bounded recursion on notation
- extensionality (optional)
- NP induction

The 1-section of PV^ω coincides with the polytime functions.

Results

Theorem (S '04)

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*

Results

Theorem (S '04)

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*

Conjecture

PT characterizes the BFF's in *all finite types*.

The conjecture holds for the intuitionistic version of PT as follows from work by Cantini.

Types and names in explicit mathematics

Types and names in explicit mathematics

- Types are collections of individuals and can have quite complicated defining properties

Types and names in explicit mathematics

- Types are collections of individuals and can have quite complicated defining properties
- Types are represented by operations or names

Types and names in explicit mathematics

- Types are collections of individuals and can have quite complicated defining properties
- Types are represented by operations or names
- Each type may have several different names or representations

Types and names in explicit mathematics

- Types are collections of individuals and can have quite complicated defining properties
- Types are represented by operations or names
- Each type may have several different names or representations
- The interplay of names and types on the level of operations witnesses the explicit character of explicit mathematics

Types and names in explicit mathematics

- Types are collections of individuals and can have quite complicated defining properties
- Types are represented by operations or names
- Each type may have several different names or representations
- The interplay of names and types on the level of operations witnesses the explicit character of explicit mathematics
- In the follwing we use a formalization of the types-and-names-paradigm due to Jäger

The language of types and names

The language $\mathcal{L}_\mathsf{T}$ is a two-sorted language extending $\mathcal{L}$ by

The language of types and names

The language $\mathcal{L}_\mathsf{T}$ is a two-sorted language extending $\mathcal{L}$ by

- type variables $U, V, W, X, Y, Z, \ldots$
- 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 language of types and names

The language $\mathcal{L}_\mathsf{T}$ is a two-sorted language extending $\mathcal{L}$ by

- type variables $U, V, W, X, Y, Z, \ldots$
- 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* $A, B, C, \ldots$ of $\mathcal{L}_\mathsf{T}$ are built from the atomic formulas of $\mathcal{L}$ 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.

Ontological axioms

We use the following abbreviations:

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

Ontological axioms

We use the following abbreviations:

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

Ontological axioms (explicit representation and extensionality)

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

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

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

The system PET

Define $\mathsf{W}_a(x) \ :=\ \mathsf{W}(x) \wedge x \leq a$.

Type existence axioms

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

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

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

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

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

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

The system PET (continued)

Type induction on W

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

Definition (The theory PET)

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

- the ontological axioms
- the above type existence axioms
- type induction on W

Proof-theoretic strength of PET

Let PT^- be PT without universal quantifiers in induction formulas.

Theorem (Spescha, S. '08)

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

The lower bounds use an involved embedding of PT^- into PET.

The upper bounds proceed via a model-theoretic argument.

Additional principles I

Totality, extensionality, choice

Totality of application:

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

Extensionality of operations:

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

Axiom of choice:

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

where A[x,y] is a positive elementary formula.

Additional principles II

Uniformity, universal quantification

Uniformity principle (Cantini)

(UP) $\forall x(\exists y \in \mathsf{W})A[x,y] \rightarrow (\exists y \in \mathsf{W})(\forall x)A[x,y]$

where $A[x,y]$ is positive elementary.

Universal quantification:

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

Results

Theorem

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 Join axiom

The Join axioms are given by the following assertions (**J.1**) and (**J.2**):

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

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

where $\Sigma[f, a, x]$ is the formula

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

Conjecture

Join does not increase the proof-theoretic strength of PET.

Extensions of PT by a partial truth predicate

Andrea Cantini has studied various extensions of 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.

Truth axioms

New (atomic) formula: $\mathsf{T}(t)$

$x \in a \;:=\; \mathsf{T}(ax)$

$\{x : A\} \;:=\; \lambda x.[A] \qquad ([A]$ term with the same free variables as $A)$

Truth axioms

$$
\begin{aligned}
\mathsf{T}[A] &\leftrightarrow A && (A \equiv (x = y), x \in \mathsf{W}) \\
\mathsf{T}(x \dot{\wedge} y) &\leftrightarrow \mathsf{T}(x) \wedge \mathsf{T}(y) \\
\mathsf{T}(x \dot{\vee} y) &\leftrightarrow \mathsf{T}(x) \vee \mathsf{T}(y) \\
\mathsf{T}(\dot{\forall} f) &\leftrightarrow \forall x \mathsf{T}(fx) \\
\mathsf{T}(\dot{\exists} f) &\leftrightarrow \exists x \mathsf{T}(fx)
\end{aligned}
$$

Choice and uniformity

Positive choice and uniformity in the truth theoretic setting:

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

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

Theorem (Cantini)

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

Proof methods used by Cantini: internal forcing semantics, non-standard variants of realizability, cut elimination.

Addendum: Positive induction

Let $(\mathsf{Pos\text{-}I_W})$ denote the schema of induction on W for positive formulas.

Theorem (Cantini)

$\boldsymbol{\tau}(\mathsf{B} + (\mathsf{Pos\text{-}I_W}))$ *coincides with the primitive recursive functions.*

Cantini's original proof uses a formalized asymmetric interpretation in $\mathsf{I}\Sigma_1$.

Alternatively, one can use the realizability techniques outlined in this talk.

Addendum: Positive safe induction

Andrea Cantini has also devised natural applicative systems for FPTIME that are inspired by the work of Leivant and Cook-Bellantoni in implicit computational complexity.

According to this approach, one uses two tiers (or sorts) W_0 and W_1 of binary words and allows induction over W_1 with respect to formulas which are positive and do only mention W_0.

In this way, applicative theories based on combinatory logic provide a natural basis also in the context of implicit computational complexity.

Future work

Future topics for research include:

- clarify the role of further type-theoretic principles such as join
- study theories of types and names for complexity classes other than FPTIME
- weak universes and reflection principles
- etc.

Selected References

1. S. Feferman, A language and axioms for explicit mathematics, Algebra and Logic, LNM 450, 1975
2. G. Jäger, Induction in the elementary theory of types and names, Computer Science Logic '87, LNCS 329, 1988
3. A. Cantini, Choice and uniformity in weak applicative theories, Logic Colloquium '01, LNL 20, ASL, 2005
4. A. Cantini, Polytime, combinatory logic and positive safe induction, Archive for Mathematical Logic 41, 2002
5. T. Strahm, Theories with self-application and computational complexity, Information and Computation 185, 2003
6. T. Strahm, A proof-theoretic characterization of the basic feasible functionals, Theoretical Computer Science 329, 2004
7. D. Spescha, T. Strahm, Elementary explicit types and polynomial time operations, Mathematical Logic Quarterly (to appear)