On the proof theory of type 2 functionals

Thomas Strahm

Institut für Informatik und angewandte Mathematik
Universität Bern

Oberwolfach, March 2005

1. The formal framework of applicative theories
2. The non-constructive μ-operator and the E_1 functional
3. Applicative theories based on primitive recursive operations
4. Bounded applicative theories and higher type feasibility

Applicative theories

- Operational core of Feferman's systems of explicit mathematics (Feferman '75)

Applicative theories

- Operational core of Feferman's systems of explicit mathematics (Feferman '75)
- Untyped universe of operations or rules, which can freely be applied to each other: self-application is meaningful, though not necessarily total

Applicative theories

- Operational core of Feferman's systems of explicit mathematics (Feferman '75)
- Untyped universe of operations or rules, which can freely be applied to each other: self-application is meaningful, though not necessarily total
- Natural setting for studying notions of abstract computability, especially from a proof-theoretic perspective

Aim of this talk

- Discussing various aspects relating to the proof theory of type 2 functionals in Feferman-style applicative theories

Aim of this talk

- Discussing various aspects relating to the proof theory of type 2 functionals in Feferman-style applicative theories
- Functionals from generalized recursion theory (E_0 and E_1)

Aim of this talk

- Discussing various aspects relating to the proof theory of type 2 functionals in Feferman-style applicative theories
- Functionals from generalized recursion theory (E_0 and E_1)
- Addressing the proof theory of these functionals also on the basis of Schlüter's partial enumerative algebra

Aim of this talk

- Discussing various aspects relating to the proof theory of type 2 functionals in Feferman-style applicative theories
- Functionals from generalized recursion theory (E_0 and E_1)
- Addressing the proof theory of these functionals also on the basis of Schlüter's partial enumerative algebra
- Provability ot type 2 functionals in weak applicative frameworks, thus discussing questions about type two feasibility

The basic language of applicative theories

The basic language of applicative theories

L is a first order language for the logic of partial terms:

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

Notation

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- 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 language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, s_N, p_N, r_N, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

Notation

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

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, $\mathsf{p_0}$, $\mathsf{p_1}$, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \;:=\; (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \;:=\; t_1{\downarrow} \vee t_2{\downarrow} \;\rightarrow\; t_1 = t_2$
- $t \in \mathsf{N} \;:=\; \mathsf{N}(t)$

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \ := \ (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \ := \ t_1{\downarrow} \vee t_2{\downarrow} \ \rightarrow \ t_1 = t_2$
- $t \in \mathsf{N} \ := \ \mathsf{N}(t)$
- $t \in \mathsf{N}^k \rightarrow \mathsf{N} \ := \ (\forall x_1 \ldots x_k \in \mathsf{N}) t x_1 \ldots x_k \in \mathsf{N}$

The basic language of applicative theories

L is a first order language for the logic of partial terms:

- constants k, s, p, p_0, p_1, 0, $\mathsf{s_N}$, $\mathsf{p_N}$, $\mathsf{r_N}$, ...
- relation symbols $=$, $\downarrow$, N
- arbitrary term application $\circ$

Notation

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

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).

The non-logical axioms of BON include:

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).
The non-logical axioms of BON include:

- partial combinatory algebra:

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

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).
The non-logical axioms of BON 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

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).
The non-logical axioms of BON 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 natural numbers N with 0, $\mathsf{s_N}$ (successor) and $\mathsf{p_N}$ (predecessor)

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman).
The non-logical axioms of BON 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 natural numbers N with 0, $\mathsf{s_N}$ (successor) and $\mathsf{p_N}$ (predecessor)
- definition by numerical cases $\mathsf{d_N}$ on N

The basic theory of operations and numbers BON

The logic of BON is the logic of partial terms (Beeson/Feferman). The non-logical axioms of BON 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 natural numbers N with 0, $\mathsf{s_N}$ (successor) and $\mathsf{p_N}$ (predecessor)
- definition by numerical cases $\mathsf{d_N}$ on N
- primitive recursion $\mathsf{r_N}$ on N (optional)

Additional axioms and sets of natural numbers

Additional axioms and sets of natural numbers

Totality (Tot)

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

Extensionality (Ext)

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

Additional axioms and sets of natural numbers

Totality (Tot)

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

Extensionality (Ext)

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

Sets of natural numbers

Additional axioms and sets of natural numbers

Totality (Tot)

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

Extensionality (Ext)

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

Sets of natural numbers

are represented via their total characteristic functions:

$$f \in \mathcal{P}(\mathsf{N}) \;\Leftrightarrow\; (\forall x \in \mathsf{N})(fx = 0 \vee fx = 1)$$

Consequences of the partial combinatory algebra axioms

Consequences of the partial combinatory algebra axioms

As usual in untyped applicative settings we have:

Consequences of the partial combinatory algebra axioms

As usual in untyped applicative settings we have:

- explicit definitions (λ-abstraction)

Consequences of the partial combinatory algebra axioms

As usual in untyped applicative settings we have:

- explicit definitions (λ-abstraction)
- recursion theorem

$$\mathsf{fix} f\downarrow \;\wedge\; (\forall x)(\mathsf{fix} f x \simeq f(\mathsf{fix} f)x)$$

Induction principles on N

Induction principles on N

Set induction on N (S-I_N)

$$f \in \mathcal{P}(\mathsf{N}) \wedge f0 = 0 \wedge (\forall x \in \mathsf{N})(fx = 0 \rightarrow f(x') = 0) \\ \rightarrow (\forall x \in \mathsf{N})(fx = 0)$$

Induction principles on N

Set induction on N (S-$\mathsf{I_N}$)

$$f \in \mathcal{P}(\mathsf{N}) \wedge f0 = 0 \wedge (\forall x \in \mathsf{N})(fx = 0 \rightarrow f(x') = 0) \\ \rightarrow (\forall x \in \mathsf{N})(fx = 0)$$

N induction on N (N-$\mathsf{I_N}$)

Induction principles on N

Set induction on N (S-I_N)

$$f \in \mathcal{P}(\mathsf{N}) \wedge f0 = 0 \wedge (\forall x \in \mathsf{N})(fx = 0 \rightarrow f(x') = 0) \\ \rightarrow (\forall x \in \mathsf{N})(fx = 0)$$

N induction on N (N-I_N)

$$f0 \in \mathsf{N} \wedge (\forall x \in \mathsf{N})(fx \in \mathsf{N} \rightarrow f(x') \in \mathsf{N}) \rightarrow (\forall x \in \mathsf{N})(fx \in \mathsf{N})$$

Induction principles on N

Set induction on N (S-I_N)

$$f \in \mathcal{P}(\mathsf{N}) \wedge f0 = 0 \wedge (\forall x \in \mathsf{N})(fx = 0 \rightarrow f(x') = 0) \\ \rightarrow (\forall x \in \mathsf{N})(fx = 0)$$

N induction on N (N-I_N)

$$f0 \in \mathsf{N} \wedge (\forall x \in \mathsf{N})(fx \in \mathsf{N} \rightarrow f(x') \in \mathsf{N}) \rightarrow (\forall x \in \mathsf{N})(fx \in \mathsf{N})$$

Formula induction on N (L-I_N)

Induction principles on N

Set induction on N (S-$\mathsf{I_N}$)

$$f \in \mathcal{P}(\mathsf{N}) \wedge f0 = 0 \wedge (\forall x \in \mathsf{N})(fx = 0 \rightarrow f(x') = 0) \\ \rightarrow (\forall x \in \mathsf{N})(fx = 0)$$

N induction on N (N-$\mathsf{I_N}$)

$$f0 \in \mathsf{N} \wedge (\forall x \in \mathsf{N})(fx \in \mathsf{N} \rightarrow f(x') \in \mathsf{N}) \rightarrow (\forall x \in \mathsf{N})(fx \in \mathsf{N})$$

Formula induction on N ($\mathcal{L}$-$\mathsf{I_N}$)

The full induction schema.

A folklore theorem

Theorem

1. $\mathsf{BON} + (\mathsf{S}\text{-}\mathsf{I_N}) \equiv \mathsf{BON} + (\mathsf{N}\text{-}\mathsf{I_N}) \equiv \mathsf{PRA}$.
2. $\mathsf{BON} + (\mathrm{L}\text{-}\mathsf{I_N}) \equiv \mathsf{PA}$.

The type two functionals E_0 and E_1

The type two functionals E_0 and E_1

$\alpha, \beta, \gamma, \dots : \mathbb{N} \to \mathbb{N}$.

The type two functionals E_0 and E_1

$\alpha, \beta, \gamma, \dots : \mathbb{N} \to \mathbb{N}$.

- The E_0 functional:

$$\mathsf{E}_0(\alpha) = \begin{cases} 0 & \exists n \alpha(n) = 0, \\ 1 & \text{else} \end{cases}$$

The type two functionals E_0 and E_1

$\alpha, \beta, \gamma, \dots : \mathbb{N} \to \mathbb{N}$.

- The E_0 functional:

$$\mathsf{E}_0(\alpha) = \begin{cases} 0 & \exists n \alpha(n) = 0, \\ 1 & \text{else} \end{cases}$$

- The E_1 functional:

$$\mathsf{E}_1(\alpha) = \begin{cases} 0 & \exists \beta \forall n \alpha(\langle \beta(n+1), \beta(n) \rangle) = 0, \\ 1 & \text{else} \end{cases}$$

Some facts from recursion theory

I	$\omega_1[I]$	recursive in I	r.e. in I

Some facts from recursion theory

I	$\omega_1[I]$	recursive in I	r.e. in I
E_0	ω_1	$L_{\omega_1} \cap \mathcal{P}(\mathbb{N})$	Σ_1 on L_{ω_1}

Some facts from recursion theory

I	$\omega_1[I]$	recursive in I	r.e. in I
E_0	ω_1	$L_{\omega_1} \cap \mathcal{P}(\mathbb{N})$	Σ_1 on L_{ω_1}
E_1	i_0	$L_{i_0} \cap \mathcal{P}(\mathbb{N})$	Σ_1 on L_{i_0}

Formalizing the functionals in the applicative setting

Formalizing the functionals in the applicative setting

The μ functional

- $(\mu.1)\quad f \in (\mathsf{N} \to \mathsf{N}) \leftrightarrow \mu f \in \mathsf{N}$,

Formalizing the functionals in the applicative setting

The μ functional

- $(\mu.1) \quad f \in (\mathsf{N} \to \mathsf{N}) \leftrightarrow \mu f \in \mathsf{N}$,
- $(\mu.2) \quad f \in (\mathsf{N} \to \mathsf{N}) \wedge (\exists x \in \mathsf{N})(fx = 0) \to f(\mu f) = 0$.

Formalizing the functionals in the applicative setting

The μ functional

- $(\mu.1)$ $\quad f \in (\mathsf{N} \to \mathsf{N}) \leftrightarrow \mu f \in \mathsf{N}$,
- $(\mu.2)$ $\quad f \in (\mathsf{N} \to \mathsf{N}) \wedge (\exists x \in \mathsf{N})(fx = 0) \to f(\mu f) = 0$.

Formalizing the functionals in the applicative setting

The μ functional

- $(\mu.1) \quad f \in (\mathsf{N} \to \mathsf{N}) \leftrightarrow \mu f \in \mathsf{N}$,
- $(\mu.2) \quad f \in (\mathsf{N} \to \mathsf{N}) \wedge (\exists x \in \mathsf{N})(fx = 0) \to f(\mu f) = 0$.

The E_1 functional

- $(\mathsf{E}_1.1) \quad f \in (\mathsf{N}^2 \to \mathsf{N}) \leftrightarrow \mathsf{E}_1 f \in \mathsf{N}$,

Formalizing the functionals in the applicative setting

The μ functional

- $(\mu.1)\quad f \in (\mathsf{N} \to \mathsf{N}) \leftrightarrow \mu f \in \mathsf{N}$,
- $(\mu.2)\quad f \in (\mathsf{N} \to \mathsf{N}) \wedge (\exists x \in \mathsf{N})(fx = 0) \to f(\mu f) = 0$.

The E_1 functional

- $(\mathsf{E}_1.1)\quad f \in (\mathsf{N}^2 \to \mathsf{N}) \leftrightarrow \mathsf{E}_1 f \in \mathsf{N}$,
- $(\mathsf{E}_1.2)\quad f \in (\mathsf{N}^2 \to \mathsf{N}) \to$

$$[(\exists g \in \mathsf{N} \to \mathsf{N})(\forall x \in \mathsf{N})(f(gx')(gx) = 0) \leftrightarrow \mathsf{E}_1 f = 0].$$

Proof theory of μ (or E_0)

Theorem (Feferman, Jäger, S.)

1. $\mathsf{BON}(\mu) + (\mathsf{S}\text{-}\mathsf{I_N}) \equiv \mathsf{PA}$,
2. $\mathsf{BON}(\mu) + (\mathsf{N}\text{-}\mathsf{I_N}) \equiv (\Delta^1_1\text{-}\mathit{CR})$,
3. $\mathsf{BON}(\mu) + (\mathrm{L}\text{-}\mathsf{I_N}) \equiv (\Delta^1_1\text{-}\mathit{CA})$.

Furthermore, all these equivalences also hold in the presence of (Tot) *and* (Ext)*.*

Proof theory of E_1

Theorem (Jäger, S.)

1. $\mathsf{BON}(\mu, \mathsf{E}_1) + (\mathsf{S}\text{-}\mathsf{I_N}) \equiv (\Delta^1_2\text{-}CA)\restriction$,
2. $\mathsf{BON}(\mu, \mathsf{E}_1) + (\mathsf{N}\text{-}\mathsf{I_N}) \equiv (\Delta^1_2\text{-}CR)$,
3. $\mathsf{BON}(\mu, \mathsf{E}_1) + (\mathrm{L}\text{-}\mathsf{I_N}) \equiv (\Delta^1_2\text{-}CA)$.

Furthermore, all these equivalences also hold in the presence of (Tot) *and* (Ext).

1. The formal framework of applicative theories
2. The non-constructive μ-operator and the E_1 functional
3. Applicative theories based on primitive recursive operations
4. Bounded applicative theories and higher type feasibility

Schlüter's partial enumerative algebra

- Define a weakening of a partial combinatory algebra for enumerated classes of functions

Schlüter's partial enumerative algebra

- Define a weakening of a partial combinatory algebra for enumerated classes of functions
- It is not necessarily assumed that the enumerating function itself belongs to that class of functions

Schlüter's partial enumerative algebra

- Define a weakening of a partial combinatory algebra for enumerated classes of functions
- It is not necessarily assumed that the enumerating function itself belongs to that class of functions
- Standard interpretation in the primitive recursive indices possible

Schlüter's partial enumerative algebra

- Define a weakening of a partial combinatory algebra for enumerated classes of functions
- It is not necessarily assumed that the enumerating function itself belongs to that class of functions
- Standard interpretation in the primitive recursive indices possible
- Aim: study the proof theory of this new algebra augmented with the type two functionals μ and E_1

The theory PRON

The theory PRON

PRON is obtained from BON by replacing the axioms for a partial combinatory algebra by the following three axioms, using two new combinators a, b and i:

- $\mathsf{k}xy = x \qquad \mathsf{i}x = x$

The theory PRON

PRON is obtained from BON by replacing the axioms for a partial combinatory algebra by the following three axioms, using two new combinators a, b and i:

- $\mathsf{k}xy = x \qquad \mathsf{i}x = x$
- $\mathsf{p}_0\langle x, y\rangle = x \wedge \mathsf{p}_1\langle x, y\rangle = y$

The theory PRON

PRON is obtained from BON by replacing the axioms for a partial combinatory algebra by the following three axioms, using two new combinators a, b and i:

- $\mathsf{k}xy = x \qquad \mathsf{i}x = x$
- $\mathsf{p}_0\langle x, y\rangle = x \wedge \mathsf{p}_1\langle x, y\rangle = y$
- $\mathsf{a}\langle x, y\rangle\downarrow \wedge \mathsf{a}\langle x, y\rangle z \simeq \langle xz, yz\rangle$

The theory PRON

PRON is obtained from BON by replacing the axioms for a partial combinatory algebra by the following three axioms, using two new combinators a, b and i:

- $\mathsf{k}xy = x \qquad \mathsf{i}x = x$
- $\mathsf{p}_0\langle x, y\rangle = x \wedge \mathsf{p}_1\langle x, y\rangle = y$
- $\mathsf{a}\langle x, y\rangle\downarrow \wedge \mathsf{a}\langle x, y\rangle z \simeq \langle xz, yz\rangle$
- $\mathsf{b}\langle x, y\rangle\downarrow \wedge \mathsf{b}\langle x, y\rangle z \simeq x(yz)$

Explicit definitions in PRON

Explicit definitions in PRON

Define a variable x to be in argument position in a term t, if x or something computed out of it is not applied to anything else. In this case we get λ-abstraction as usual:

Explicit definitions in PRON

Define a variable x to be in argument position in a term t, if x or something computed out of it is not applied to anything else. In this case we get λ-abstraction as usual:

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

Explicit definitions in PRON

Define a variable x to be in argument position in a term t, if x or something computed out of it is not applied to anything else. In this case we get λ-abstraction as usual:

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

- **Not allowed:** $\lambda z.xz(yz), \ \lambda x.(x)_0(x)_1$

Explicit definitions in PRON

Define a variable x to be in argument position in a term t, if x or something computed out of it is not applied to anything else. In this case we get λ-abstraction as usual:

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

- **Not allowed:** $\lambda z.xz(yz), \ \lambda x.(x)_0(x)_1$
- **Allowed:** $\lambda z.\langle xz, yz\rangle$

Proof theory of μ (or E_0) on the basis of PRON

Theorem (Steiner, S.)

1. $\mathsf{PRON}(\mu) + (\mathsf{S}\text{-}\mathsf{I_N}) \equiv \mathsf{PA}$,
2. $\mathsf{PRON}(\mu) + (\mathrm{L}\text{-}\mathsf{I_N}) \equiv (\Pi^0_1\text{-}\mathsf{CA})$.

Proof theory of E_1 on the basis of PRON

Theorem (Steiner, S.)

1. $\mathsf{PRON}(\mu, \mathsf{E}_1) + (\mathsf{S}\text{-}\mathsf{I}_\mathsf{N}) \equiv (\Pi^1_1\text{-}\mathsf{CA})\!\upharpoonright$,
2. $\mathsf{PRON}(\mu, \mathsf{E}_1) + (\mathrm{L}\text{-}\mathsf{I}_\mathsf{N}) \equiv (\Pi^1_1\text{-}\mathsf{CA})$.

Bounded applicative theories and higher type functionals

General program

- Set up and study of bounded Feferman-style applicative theories capturing various complexity classes in a uniform way

Bounded applicative theories and higher type functionals

General program

- Set up and study of bounded Feferman-style applicative theories capturing various complexity classes in a uniform way
- Exhibit relationship to systems of bounded arithmetic, higher type functionals

Bounded applicative theories and higher type functionals

General program

- Set up and study of bounded Feferman-style applicative theories capturing various complexity classes in a uniform way
- Exhibit relationship to systems of bounded arithmetic, higher type functionals
- Higher types arise naturally in an untyped applicative setting!

Bounded applicative theories and higher type functionals

General program

- Set up and study of bounded Feferman-style applicative theories capturing various complexity classes in a uniform way
- Exhibit relationship to systems of bounded arithmetic, higher type functionals
- Higher types arise naturally in an untyped applicative setting!

Bounded applicative theories and higher type functionals

General program

- Set up and study of bounded Feferman-style applicative theories capturing various complexity classes in a uniform way
- Exhibit relationship to systems of bounded arithmetic, higher type functionals
- Higher types arise naturally in an untyped applicative setting!

In the sequel: Presentation of an applicative theory PT which characterizes the type 2 basic feasible functionals

The language of PT

L_W is a first order language for the logic of partial terms:

The language of PT

L_{W} 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}, $*$, $\times$

The language of PT

L_W 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, $*$, $\times$
- relation symbols $=$, $\downarrow$, W

The language of PT

L_W 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, $*$, $\times$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Basic axioms of PT

Basic axioms of PT

PT is based on the classical logic of partial terms. Its non-logical axioms include:

- partial combinatory algebra

Basic axioms of PT

PT is based on the classical logic of partial terms. Its non-logical axioms include:

- partial combinatory algebra
- pairing and projections

Basic axioms of PT

PT is based on the classical logic of partial terms. Its non-logical axioms include:

- partial combinatory algebra
- pairing and projections
- defining axioms for the binary words W with ϵ and the binary successors s_0 and s_1, predecessor d_W and definition by cases d_W on W

Basic axioms of PT

PT is based on the classical logic of partial terms. Its non-logical axioms include:

- partial combinatory algebra
- pairing and projections
- defining axioms for the binary words W with ϵ and the binary successors s_0 and s_1, predecessor d_W and definition by cases d_W on W
- word concatenation $*$ and word multiplication $\times$

A natural induction principle for PT

A natural induction principle for PT

Σ^b_{W}-formulas

A natural induction principle for PT

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

A natural induction principle for PT

Σ^b_{W}-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Σ^b_{W}-induction on W

A natural induction principle for PT

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}-induction on W

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

PT characterizes the type 2 basic feasible functionals

Theorem (S)

The provably total type two functionals of PT *coincide with* BFF_2*, the basic feasible functionals of type two.*

Final remark and an open problem

Final remark and an open problem

PT and PV^ω

Final remark and an open problem

PT and PV^ω

Indeed, the basic feasible functionals in all finite types are provably total in PT.

Final remark and an open problem

PT and PV^ω

Indeed, the basic feasible functionals in all finite types are provably total in PT.

Question

Is every provably total functional of type ≥ 3 of PT basic feasible?

Final remark and an open problem

PT and PV^ω

Indeed, the basic feasible functionals in all finite types are provably total in PT.

Question

Is every provably total functional of type ≥ 3 of PT basic feasible?

Conjecture: Yes