Weak theories of truth and explicit mathematics

Thomas Strahm (joint work with Sebastian Eberhard)

Institut für Informatik und angewandte Mathematik, Universität Bern

Oxford, Sep 19, 2011

General aims of this talk

In this talk we will discuss

- weak theories of truth over combinatory logic
- weak systems of Feferman's explicit mathematics
- the relationship between the two formalisms
- we consider two truth theories T_{PR} and T_{PT} of primitive recursive and polynomial time strength, respectively
- T_{PT} is a novel abstract truth-theoretic framework which is able to interpret feasible subsystems of explicit mathematics
- the proof that T_{PT} is feasible is non-trivial and due to Sebastian Eberhard (see his talk tomorrow)

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 and object-oriented programming languages
- universes and higher reflection principles
- formal proof-theoretic framework for abstract computations from ordinary and generalized recursion theory

Theories of truth over combinarory logic

The truth theories over combinatory logic relevant to this talk have various roots:

- illative combinatory logic (Curry, Fitch)
- Frege structures (Scott, Aczel, Flagg and Myhill)
- Kripke-Feferman style axiomatizations of truth over combinatory algebras
- Intensively studied by Cantini (see e.g. his monograph *Logical frameworks for truth and abstraction*) and Kahle (see e.g. his Habilitationsschrift *The applicative realm*)

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
- $*$: 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
- $*$: 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

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

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 L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

The basic applicative language L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

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

The basic applicative language L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \;:=\; (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \;:=\; t_1{\downarrow} \vee t_2{\downarrow} \to t_1 = t_2$

The basic applicative language L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

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

The basic applicative language L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

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

The basic applicative language L

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}, $\mathsf{c}_{\subseteq}$, $*$, $\times \ldots$
- relation symbols $=$, $\downarrow$, W
- arbitrary term application $\circ$

Notation

- $t_1 t_2 \ldots t_n \;:=\; (\ldots(t_1 \circ t_2) \circ \cdots \circ t_n)$
- $t_1 \simeq t_2 \;:=\; t_1{\downarrow} \vee t_2{\downarrow} \to t_1 = t_2$
- $t \in \mathsf{W} \;:=\; \mathsf{W}(t)$
- $t : \mathsf{W}^k \to \mathsf{W} \;:=\; (\forall x_1 \ldots x_k \in \mathsf{W}) t x_1 \ldots x_k \in \mathsf{W}$
- $s \leq t \;:=\; 1{\times}s \subseteq 1{\times}t$

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 and the predecessor p_W.
- definition by cases d_W on W
- initial subword relation $\mathsf{c}_\subseteq$
- word concatenation $*$ and word multiplication $\times$

Consequences of the partial combinatory algebra axioms

As usual in untyped applicative settings we have:

Lemma (Explicit definitions and fixed points)

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

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

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

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

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 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 $\overline{w}$
- Note that $\mathcal{M}(\lambda\eta)$ satiesfies totality of application (**Tot**) and extensionality of operations (**Ext**)

Natural induction principles

Natural induction principles

Σ^b_W-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Natural induction principles

Σ^b_W-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Σ^b_W notation induction on W, $(\Sigma^b_W\text{-I}_W)$

$$f : W \to W \wedge A(\epsilon) \wedge (\forall x \in W)(A(x) \to A(s_0 x) \wedge A(s_1 x)) \to (\forall x \in W)A(x)$$

Natural induction principles

Σ^b_W-formulas

Formulas $A(x)$ of the form

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

for B positive and W-free

Σ^b_W notation induction on W, $(\Sigma^b_W\text{-I}_W)$

$$f : W \to W \wedge A(\epsilon) \wedge (\forall x \in W)(A(x) \to A(s_0 x) \wedge A(s_1 x)) \to (\forall x \in W)A(x)$$

Positive induction on W, (Pos-I_W)

Induction on W for arbitrary positive formulas.

Provably total functions

Definition

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

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

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

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

Proof-theoretic results I

The two systems PT and PR

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

$$\mathsf{PR} := \mathsf{B} + (\mathsf{Pos}\text{-}\mathsf{I}_{\mathsf{W}})$$

Proof-theoretic results I

The two systems PT and PR

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

$$\mathsf{PR} := \mathsf{B} + (\mathsf{Pos}\text{-}\mathsf{I}_{\mathsf{W}})$$

Theorem (Cantini)

$\tau(\mathsf{PR})$ *equals the class of primitive recursive functions.*

Proof-theoretic results I

The two systems PT and PR

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

$$\mathsf{PR} := \mathsf{B} + (\mathsf{Pos}\text{-}\mathsf{I}_{\mathsf{W}})$$

Theorem (Cantini)

$\tau(\mathsf{PR})$ *equals the class of primitive recursive functions.*

Theorem (S.)

$\tau(\mathsf{PT})$ *equals the class of polynomial time computable functions.*

u^b
UNIVERSITÄT BERN

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 following we use a formalization of the types and names paradigm due to Jäger

The language of types and names

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

The language of types and names

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

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

The language of types and names

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

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

The *formulas* $A, B, C, \ldots$ of $\mathbb{L}$ are built from the atomic formulas of L as well as from 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:

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

Ontological axioms

We use the following abbreviations:

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

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 EPC

Type existence axioms

(w) $\Re(\mathsf{w}) \wedge \forall x(x \mathrel{\dot\in} \mathsf{w} \leftrightarrow \mathsf{W}(x))$

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

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

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

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

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

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

The system EPC (continued)

Type induction on W

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

Definition (The theory EPC)

EPC is the extension of the first-order applicative theory B by

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

The system PET

PET is a subsystem of EPC where it is no longer claimed that the class W of words forms a type.

The system PET

PET is a subsystem of EPC where it is no longer claimed that the class W of words forms a type.

Definition

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

The system PET

PET is a subsystem of EPC where it is no longer claimed that the class W of words forms a type.

Definition

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

Only initial segments of W define types:

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

All other axioms of PET are identical to the ones of EPC.

The Join axiom

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

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

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

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

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

Let us write EPCJ and PETJ for the extension of EPC and PET by the join principle.

Proof-theoretic results II

Theorem (Cantini)

τ(EPCJ) *equals the class of primitive recursive functions.*

Proof-theoretic results II

Theorem (Cantini)

$\tau(\mathsf{EPCJ})$ *equals the class of primitive recursive functions.*

Theorem (Spescha, S.)

$\tau(\mathsf{PETJ}^i)$ *equals the class of polynomial time computable functions.*

Proof-theoretic results II

Theorem (Cantini)

$\tau(\mathsf{EPCJ})$ *equals the class of primitive recursive functions.*

Theorem (Specha, S.)

$\tau(\mathsf{PETJ}^i)$ *equals the class of polynomial time computable functions.*

Theorem (Probst)

$\tau(\mathsf{PETJ})$ *equals the class of polynomial time computable functions.*

The language L_T of positive truth

The (first order) language L_T is an extension of the language L by

The language L_T of positive truth

The (first order) language L_T is an extension of the language L by

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

The language L_T of positive truth

The (first order) language L_T is an extension of the language L by

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

The new constants allow only the coding of positive formulas since we do not add a constant $\dot{\neg}$ for negation. As usual, we will use infix notation for $\dot{=}$, $\dot{\wedge}$, and $\dot{\vee}$.

The language L_T of positive truth

The (first order) language L_T is an extension of the language L by

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

The new constants allow only the coding of positive formulas since we do not add a constant $\dot{\neg}$ for negation. As usual, we will use infix notation for $\dot{=}$, $\dot{\wedge}$, and $\dot{\vee}$.

The formulas of L_T are built as expected, where for each term t, $\mathsf{T}(t)$ is a new atomic formula.

The truth theory T_{PR}

Positive truth axioms

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

The truth theory T_{PR} (continued)

Truth induction on W

$$\mathsf{T}(r\epsilon) \wedge (\forall x \in \mathsf{W})(\mathsf{T}(rx) \to \mathsf{T}(r(\mathsf{s}_0 x)) \wedge \mathsf{T}(r(\mathsf{s}_1 x))) \to (\forall x \in \mathsf{W})\mathsf{T}(rx)$$

Definition (The theory T_{PR})

T_{PR} is the extension of the first-order applicative theory B by

- totality of application
- the above truth axioms
- truth induction on W

The truth theory T_{PT}

T_{PT} is a subsystem of T_{PR} where the truth predicate can only reflect initial segments of the predicate W.

The truth theory T_{PT}

T_{PT} is a subsystem of T_{PR} where the truth predicate can only reflect initial segments of the predicate W.

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

The truth theory T_{PT}

T_{PT} is a subsystem of T_{PR} where the truth predicate can only reflect initial segments of the predicate W.

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

Only initial segments of W are reflected by T:

$$a \in \mathsf{W} \rightarrow (\mathsf{T}(\dot{\mathsf{W}}ax) \leftrightarrow \mathsf{W}_a(x))$$

All other axioms of T_{PT} are identical to the ones of T_{PR}.

Proof-theoretic results III

Theorem (Cantini)

$\tau(\mathsf{T}_{\mathsf{PR}})$ equals the class of primitive recursive functions.

Proof-theoretic results III

Theorem (Cantini)

$\tau(\mathsf{T}_{\mathsf{PR}})$ *equals the class of primitive recursive functions.*

Theorem (Eberhard)

$\tau(\mathsf{T}_{\mathsf{PT}})$ *equals the class of polynomial time computable functions.*

Embedding explicit mathematics into truth theories

Embedding explicit mathematics into truth theories

- The translation of EPCJ and PETJ into T_{PR} and T_{PT}, respectively, works very easily. Basically, define $\star$ in such a way that

$$(s \mathrel{\dot\in} t)^\star \qquad \text{is simply} \qquad \mathsf{T}(t^\star s^\star)$$

Embedding explicit mathematics into truth theories

- The translation of EPCJ and PETJ into T_{PR} and T_{PT}, respectively, works very easily. Basically, define $\star$ in such a way that

$$(s \mathrel{\dot\in} t)^\star \qquad \text{is simply} \qquad \mathsf{T}(t^\star s^\star)$$

- The type constructors are interpreted by terms of L_{T} which embody their membership conditions

Embedding explicit mathematics into truth theories

- The translation of EPCJ and PETJ into T_{PR} and T_{PT}, respectively, works very easily. Basically, define $\star$ in such a way that

$$(s \dot{\in} t)^\star \qquad \text{is simply} \qquad \mathsf{T}(t^\star s^\star)$$

- The type constructors are interpreted by terms of L_{T} which embody their membership conditions
- Here we implicitly assume a first-order formulation of explicit mathematics

Embedding explicit mathematics into truth theories (ctd.)

Translation of the type constructors in the case of EPCJ

$$
\begin{aligned}
\mathsf{id}^* &\equiv \lambda z.\dot{\exists}\lambda y.\ z \doteq \langle y, y\rangle \\
\mathsf{w}^* &\equiv \lambda z.\ \dot{\mathsf{W}} z \\
\mathsf{int}^* &\equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot{\wedge}} bz \\
\mathsf{un}^* &\equiv \lambda a.\lambda b.\lambda z.\ az \mathbin{\dot{\vee}} bz \\
\mathsf{inv}^* &\equiv \lambda f.\lambda a.\lambda z.\ a(fz) \\
\mathsf{dom}^* &\equiv \lambda a.\lambda z.\ \dot{\exists}\lambda y.a\langle z, y\rangle \\
\mathsf{all}^* &\equiv \lambda a.\lambda z.\ \dot{\forall}\lambda y.a\langle z, y\rangle \\
\mathsf{j}^* &\equiv \lambda f.\lambda a.\lambda z.\ \dot{\exists}\lambda x.\dot{\exists}\lambda y.z \doteq \langle x, y\rangle \mathbin{\dot{\wedge}} ax \mathbin{\dot{\wedge}} (fx)y
\end{aligned}
$$

Embedding truth theories into explicit mathematics

Embedding truth theories into explicit mathematics

- The direct embedding of weak truth theories into explicit mathematics requires additional assumptions

Embedding truth theories into explicit mathematics

- The direct embedding of weak truth theories into explicit mathematics requires additional assumptions
- Namely, we employ the existence of universes and Cantini's uniformity principle

Embedding truth theories into explicit mathematics

- The direct embedding of weak truth theories into explicit mathematics requires additional assumptions
- Namely, we employ the existence of universes and Cantini's uniformity principle
- These principles do not raise the proof-theoretic strength of the underlying formalisms

Cantini's uniformity principle

Uniformity principle (Cantini)

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

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

Universes in explicit mathematics

Universes in explicit mathematics

- A universe in explicit mathematics is a type of names which is closed under previously recognized type existence principles

Universes in explicit mathematics

- A universe in explicit mathematics is a type of names which is closed under previously recognized type existence principles
- An EPCJ universe is closed under the type constructors of the theory EPCJ; analogously for PETJ universes

Universes in explicit mathematics

- A universe in explicit mathematics is a type of names which is closed under previously recognized type existence principles
- An EPCJ universe is closed under the type constructors of the theory EPCJ; analogously for PETJ universes
- EPCJ + U is the extension of EPCJ where it is claimed that each type is contained in an EPCJ universe; analogously for PETJ + U

Reducing T_{PR} to EPCJ + U + UP

Reducing T_{PR} to EPCJ + U + UP

- the main task is to define a type which interprets the truth predicate

Reducing T_{PR} to EPCJ + U + UP

- the main task is to define a type which interprets the truth predicate
- ω many levels suffice; we work in a universe in order to show that the truth level type τw is indeed a name for each $w \in W$

Reducing T_{PR} to EPCJ + U + UP

- the main task is to define a type which interprets the truth predicate
- ω many levels suffice; we work in a universe in order to show that the truth level type τw is indeed a name for each $w \in W$
- UP is used in order to deal with the axioms about $\dot{\forall}$

Reducing T_{PT} to PETJ + U

Reducing T_{PT} to PETJ + U

- the reduction is more delicate since we cannot collect the truth levels for all words into a type

Reducing T_{PT} to PETJ + U

- the reduction is more delicate since we cannot collect the truth levels for all words into a type
- there is an intermediate step via an asymmetric interpretation to a leveled truth theory $\mathsf{T}^{\ell}_{\mathsf{PT}}$ where the levels of the truth predicate are polynomially bounded in a specific way

Reducing T_{PT} to PETJ + U

- the reduction is more delicate since we cannot collect the truth levels for all words into a type
- there is an intermediate step via an asymmetric interpretation to a leveled truth theory $\mathsf{T}^{\ell}_{\mathsf{PT}}$ where the levels of the truth predicate are polynomially bounded in a specific way
- further $\mathsf{T}^{\ell}_{\mathsf{PT}}$ can be modeled in PETJ + U similarly as above

Reducing T_{PT} to PETJ + U

- the reduction is more delicate since we cannot collect the truth levels for all words into a type
- there is an intermediate step via an asymmetric interpretation to a leveled truth theory $\mathsf{T}^{\ell}_{\mathsf{PT}}$ where the levels of the truth predicate are polynomially bounded in a specific way
- further $\mathsf{T}^{\ell}_{\mathsf{PT}}$ can be modeled in PETJ + U similarly as above
- it should be noted that the direct proof-theoretic treatment of PETJ + U requires Eberhard's involved new realizability techniques developed for T_{PT}

Applications of T_{PT} to the unfolding program

Applications of T_{PT} to the unfolding program

- the feasible truth theory T_{PT} is also an important reference theory for our recent work on Feferman's unfolding program

Applications of T_{PT} to the unfolding program

- the feasible truth theory T_{PT} is also an important reference theory for our recent work on Feferman's unfolding program
- we are working on the unfolding of a natural schematic system FEA of feasible arithmetic

Applications of T_{PT} to the unfolding program

- the feasible truth theory T_{PT} is also an important reference theory for our recent work on Feferman's unfolding program
- we are working on the unfolding of a natural schematic system FEA of feasible arithmetic
- the system T_{PT} plays a crucial role in order to obtain proof-theoretic upper bounds for the full predicate unfolding $\mathcal{U}(\mathrm{FEA})$ of FEA

Selected References

1. A. Cantini, Proof-theoretical aspects of self-referential truth, LMPS '95, Kluwer, 1997
2. A. Cantini, Choice and uniformity in weak applicative theories, Logic Colloquium '01, LNL 20, ASL, 2005
3. S. Eberhard, A truth theory over an applicative framework of strength PT, in preparation
4. S. Eberhard, T. Strahm, Weak theories of truth and explicit mathematics, submitted.
5. D. Probst, The provably terminating operations of the subsystem PETJ of explicit mathematics, Annals of Pure and Applied Logic 162, 2011
6. D. Spescha, T. Strahm, Elementary explicit types and polynomial time operations, Mathematical Logic Quarterly 55, 2009
7. D. Spescha, T. Strahm, Realisability in weak systems of explicit mathematics, Mathematical Logic Quarterly (to appear)
8. T. Strahm, Theories with self-application and computational complexity, Information and Computation 185, 2003