

Domains VIII
Sobolev Institute of Mathematics
Novosibirsk Akademgorodok
11 - 15 September, 2007

A domain-theoretic
characterisation of strong
normalisation in the
 λ - \mathcal{R} -calculus

Ulrich Berger
Swansea University

- Introduction
- The λ - \mathcal{R} -calculus
- Domain-theoretic semantics
- Characterising strong normalisation
- Types and totality
- Applications
- Conclusion
- References

The strong normalisation problem

Given a higher type rewrite system - typically an extension of Gödel's system T by constants and rewrite rules - how can we prove strong normalisation?

Strong normalisation via totality

We define a domain model with totality such that for any rewrite system:

if all constants are total, then all terms are strongly normalising.

Advantages

1. Totality is often easy to prove (for example, the totality of Gödel's primitive recursor is proved by a trivial induction).
2. Totality is compositional, hence it can be proved for each constant separately.

Example: Gödel's system T

β -conversion, $(\lambda x.M)N \rightarrow M[N/x]$, plus

$$\begin{aligned} \mathsf{RAG} 0 &\rightarrow A \\ \mathsf{RAG} (n+1) &\rightarrow G n(\mathsf{RAG} n) \end{aligned}$$

Suppressing arguments that are not changed in the recursive call of R this simplifies to

$$\begin{aligned} \mathsf{R} 0 &\rightarrow A \\ \mathsf{R} (n+1) &\rightarrow G n(\mathsf{R} n) \end{aligned}$$

In the following examples we use this simplified notation.

Example: Spector's barrecursion

$\text{BR}(\alpha, n) \rightarrow \mathbf{if } Y \alpha < n \mathbf{ then } G(\alpha, n) \mathbf{ else } H \alpha n (\lambda x. \text{BR}(\alpha_n^x, n + 1))$

where

$$x < 0 \rightarrow \text{F}$$

$$0 < (y + 1) \rightarrow \text{T}$$

$$(x + 1) < (y + 1) \rightarrow x < y$$

and $\alpha_n^x := \lambda m. \mathbf{if } m = n \mathbf{ then } x \mathbf{ else } \alpha m$

Think of (α, n) as coding the finite sequence $[\alpha 0, \dots, \alpha (n - 1)]$.

Hence $(\alpha_n^x, n + 1)$ codes the sequence $[\alpha 0, \dots, \alpha (n - 1), x]$.

Example: Open recursion

$$\text{OR } \alpha \rightarrow Y \alpha (\lambda n, x, \beta. \mathbf{if } x \prec \alpha n \mathbf{ then OR } (\alpha_n^{x, \beta}) \mathbf{ else } 0)$$

where $\alpha_n^{x, \beta} = \lambda m. \mathbf{if } m \leq n \mathbf{ then } \alpha_n^x m \mathbf{ else } \beta m.$

Think of α as ranging over infinite sequences ordered lexicographically by \prec .

Hence $\alpha_n^{x, \beta}$, with $x \prec \alpha n$, ranges over all infinite sequences lexicographically below α .

From operational to denotational semantics

The rewrite rules we have seen are all meaningful w.r.t. a domain semantics, since they can be viewed as recursive definition.

That is, the denotational semantics of a constant is the least fixed point of the effectively continuous function explicitly defined by the rules.

Example: Nondeterministic choice

$$x \parallel y \rightarrow x$$

$$x \parallel y \rightarrow y$$

- ▶ Used by Kristiansen (CiE 2006) to characterise the nondeterministic polynomial hierarchy in terms of fragments of Gödel's T .
- ▶ What is its denotational semantics?
- ▶ Can destroy termination: extending Gödel's T by

$$f\ 0\ 1\ x \rightarrow f\ x\ x\ x$$

still terminates, but adding further \parallel yields

$$f\ 0\ 1\ (0 \parallel 1) \rightarrow f\ (0 \parallel 1)\ (0 \parallel 1)\ (0 \parallel 1) \rightarrow^2 f\ 0\ 1\ (0 \parallel 1)$$

(Toyama)

Nondeterministic denotational semantics

We interpret terms as nondeterministic values, i.e. as finite sequences of deterministic values.

The choice operator \parallel is interpreted as the concatenation operation.

From denotational semantics to strong normalisation

We characterise strong normalisation by the denotational property of having a defined value. Altogether we have:

$$\llbracket M \rrbracket \text{ total} \quad \Rightarrow \quad \llbracket M \rrbracket \neq \perp \quad \Leftrightarrow \quad M \text{ strongly normalising}$$

The main ideas

- ▶ Adequacy for PCF (Plotkin): If a closed PCF-term of base type denotes a numeral in the domain model, then it weak head reduces to that numeral.
- ▶ Characterisation of strongly normalising (pure) λ -terms by intersection types (Pottinger).
- ▶ Intersection types as a filter model of λ -terms (Barendregt, Coppo, Dezani, van Bakel).

The connection with intersection types was pointed out by Thomas Ehrhard.

Previous work

- ▶ “ $\llbracket M \rrbracket \neq \perp \Rightarrow \text{SN}(M)$ ” for deterministic rewrite systems, assuming SN for the underlying type theory (B 05).
- ▶ “ $\llbracket M \rrbracket \neq \perp \Rightarrow \text{SN}(M)$ ” for deterministic rewrite systems, unconditionally, using the “intersection types as filter models” idea (Coquand, Spiwack 06).

New in this talk:

- ▶ Nondeterminism.
- ▶ Completeness: “ $\llbracket M \rrbracket \neq \perp \Leftrightarrow \text{SN}(M)$ ”.
- ▶ Abstract domain theory instead of formal typing rules.

Terms

$\Lambda \ni M, N$	$::=$	x	variable
		c	constructor (always includes T, F)
		f	constant
		(M, N)	pair
		$\lambda x.M$	abstraction
		$M N$	application
		if (M, N)	definition by cases

Notation: **if** K **then** M **else** $N := \mathbf{if}(M, N) K$.

Rewrite systems

For every constant f we assume a list \mathcal{R}_f of *rules* of the form

$$f \vec{P} \rightarrow M$$

where

- ▶ \vec{P} is a list of *patterns*, i.e. terms built from constructors, variables and pairing, such that in \vec{P} no variable occurs more than once;
- ▶ M is a term with $FV(M) \subseteq FV(\vec{P})$;
- ▶ the length of the pattern list \vec{P} is fixed for each f (this fixed length is called the *arity* of f);
- ▶ only finitely many left hand sides are allowed to be unifiable.

Example

$$R A G 0 \quad \rightarrow \quad A$$

$$R A G (S, n) \quad \rightarrow \quad G n (R A G n)$$

R constant of arity 3

$0, S$ constructors

A, G, n variables

Reduction, $K \rightarrow K'$

Contracting a subterm of K which is not in a branch of an **if**-term, where

	contracts to	
$(\lambda x.M) N$		$M[N/x]$
if (M, N) T		M
if (M, N) F		N
$f \vec{P} \theta$		$M \theta$ ($f \vec{P} \rightarrow M$ a rule, θ a substitution)

Strong normalisation

A term M is *strongly normalising*, $\text{SN}(M)$, if there is no infinite reduction sequence

$$M \rightarrow M' \rightarrow M'' \rightarrow \dots$$

Safety

A term is *safe* if

- (1) every constant f occurs only in contexts of the form $f M_1 \dots M_k$ where k is the arity of f ,
- (2) no constructor or pair occurs as the left hand side of an application,
- (3) (inductively) all reducts are safe.

Safety is usually guaranteed by typability.

A strict reflexive Scott-domain

$$D = \mathcal{C}_\perp \oplus (D^* \otimes D^*) \oplus (D^* \overset{!}{\rightarrow} D^*)$$

\mathcal{C}_\perp	flat domain of constructors
D^*	strict finite lists (non-deterministic values)
\otimes	strict (or smash) product
\oplus	strict (or coalesced) sum
$\overset{!}{\rightarrow}$	strict function space

The elements of $D_+ := D \setminus \perp$:

c	(c a constructor)
(\mathbf{d}, \mathbf{e})	($\mathbf{d}, \mathbf{e} \in D_+^*$)
$\text{fun}(f)$	($f: D^* \rightarrow D^*$, continuous, strict, $\neq \perp$)

Some important operations

$$\begin{aligned}
 \text{app}(\text{fun}(f), \mathbf{d}) &:= f(\mathbf{d}) \\
 \text{app}(d, \mathbf{d}) &:= \perp, \text{ if } d \text{ is a pair or a constructor} \\
 \mathbf{d} \bullet \mathbf{e} &:= [\text{app}(d, \mathbf{e}) \mid d \leftarrow \mathbf{d}] \\
 \mathbf{T} \triangleright \mathbf{d} &:= \mathbf{d} \\
 \mathbf{F} \triangleright \mathbf{d} &:= []
 \end{aligned}$$

$$\text{match}_P: D^* \rightarrow (\text{FV}(P) \rightarrow D^*)^*$$

$$\begin{aligned}
 \text{match}_x(\mathbf{d}) &= [[x \mapsto \mathbf{d}]] \\
 \text{match}_c(\mathbf{d}) &= (c \in \mathbf{d}) \triangleright [\emptyset] \\
 \text{match}_{(P,Q)}(\mathbf{d}) &= [\eta \cup \eta' \mid (\mathbf{e}, \mathbf{e}') \leftarrow \mathbf{d}, \eta \leftarrow \text{match}_P(\mathbf{e}), \\
 &\quad \eta' \leftarrow \text{match}_Q(\mathbf{e}')]
 \end{aligned}$$

The value of a term: $\llbracket M \rrbracket_\eta \in D^*$

$$\llbracket x \rrbracket_\eta = \eta(x)$$

$$\llbracket c \rrbracket_- = [c]$$

$$\llbracket (M, N) \rrbracket_\eta = [(\llbracket M \rrbracket_\eta, \llbracket N \rrbracket_\eta)]$$

$$\llbracket MN \rrbracket_\eta = \llbracket M \rrbracket_\eta \bullet \llbracket N \rrbracket_\eta$$

$$\llbracket \lambda x. M \rrbracket_\eta = [\text{fun}(\lambda \mathbf{d} \in D^*. \llbracket M \rrbracket_\eta[x := \mathbf{d}])]$$

$$\llbracket \text{if}(M, N) \rrbracket_\eta = [\text{fun}(\lambda \mathbf{d} \in D^*. (\text{T} \in \mathbf{d} \triangleright \llbracket M \rrbracket_\eta) ++ (\text{F} \in \mathbf{d} \triangleright \llbracket N \rrbracket_\eta))]$$

$$\begin{aligned} \llbracket f \rrbracket_- &= [\text{fun}^k(\lambda \vec{\mathbf{d}} \in (D^*)^k. \\ &\quad \text{concat}[\llbracket M \rrbracket_\eta \mid (\vec{P} \mapsto M) \leftarrow \mathcal{R}_f, \eta \leftarrow \text{match}_{\vec{P}}(\vec{\mathbf{d}})])] \end{aligned}$$

where $\eta: \text{FV}(M) \rightarrow D^*$ and $k = \text{arity}(f)$.

The analogy with intersection types

The relation

$$\mathbf{U} \sqsubseteq \llbracket M \rrbracket_{\eta},$$

where \mathbf{U} ranges over non-deterministic defined compacts, can be defined inductively, similar to typing judgements in the intersection type calculus ($\eta \vdash M : \mathbf{U}$).

This has been carried out (without non-determinism) by Coquand and Spiwack.

Hence, “ $\llbracket M \rrbracket_{\eta} \neq \perp$ ”, which is equivalent to “ $\exists \mathbf{U} (\mathbf{U} \sqsubseteq \llbracket M \rrbracket_{\eta})$ ”, can be read as “ M is typeable”.

Strong normalisation theorem

Set $\llbracket M \rrbracket := \llbracket M \rrbracket_{\eta_0}$ where $\eta_0(x) := []$ for all variables x .

For every safe term M ,

$$\llbracket M \rrbracket \neq \perp \iff M \text{ is strongly normalising}$$

We sketch the proof of “ \Rightarrow ” (which doesn't need the safety assumption).

Reducibility candidates

A term is *simple* if it has neither of the following forms:

$c\vec{N}$, $(M_1, M_2)\vec{N}$, $\lambda x.M$, **if then M else N** ,
 $f N_1 \dots N_k$ where $k < \text{arity}(f)$.

A *reducibility candidate* is a set X of terms such that

RC1 $X \subseteq \text{SN}$.

RC2 If $M \in X$ and $M \rightarrow M'$, then $M' \in X$.

RC3 If M is simple and $\forall M' (M \rightarrow M' \Rightarrow M' \in X)$, then $M \in X$.

$X \rightarrow Y := \{M \mid \forall N (N \in X \Rightarrow MN \in Y)\}$.

$X \times Y := \{(M, N) \mid M \in X, N \in Y\} (\subseteq \Lambda)$.

RC3(X) := the closure of X under the rule **RC3** above.

Rank

$D = \lim_n D_n$, with canonical embeddings $\epsilon_n: D_n \rightarrow D$, where

$$\begin{aligned} D_0 &= \{\perp\} \\ D_{n+1} &= \mathcal{C}_\perp \oplus (D_n^* \otimes D_n^*) \oplus (D_n^* \xrightarrow{!} D_n^*) \end{aligned}$$

For compacts $U \in D \setminus \perp$ and $\mathbf{U} \in D^* \setminus \perp$ we set

$$\begin{aligned} \text{rk}(U) &:= \min\{n \mid n \in \epsilon_n(D_n)\} \\ \text{rk}(\mathbf{U}) &:= \sup\{\text{rk}(U) \mid U \in \mathbf{U}\} \end{aligned}$$

(the stage where U resp. \mathbf{U} is constructed)

Properties of $\text{rk}(U)$

- ▶ $\text{rk}(\mathbf{U}_i) < \text{rk}((\mathbf{U}_1, \mathbf{U}_2))$.
- ▶ If $F(\mathbf{d}) \neq \perp$, then
 - $\text{rk}(F(\mathbf{d})) < \text{rk}(\text{fun}(F))$ and
 - $F(\mathbf{d}) = F(\mathbf{U})$ for some $\mathbf{U} \sqsubseteq \mathbf{d}$ with $\text{rk}(\mathbf{U}) < \text{rk}(\text{fun}(F))$.

Assigning reducibility candidates to defined compacts

$$\Lambda(c) = \mathbf{RC3}(c)$$

$$\Lambda((\mathbf{U}, \mathbf{V})) = \mathbf{RC3}(\Lambda(\mathbf{U}) \times \Lambda(\mathbf{V}))$$

$$\begin{aligned} \Lambda(\text{fun}(F)) &= \bigcap \{ \Lambda(\mathbf{U}) \rightarrow \Lambda(F(\mathbf{U})) \mid \mathbf{U} \in \text{dom}(F), \text{rk}(\mathbf{U}) < \text{rk}(\text{fun}(F)) \} \\ &= \bigcap \{ \Lambda(\mathbf{U}) \rightarrow \Lambda(F(\mathbf{U})) \mid \mathbf{U} \in \text{dom}(F) \} \end{aligned}$$

$$\Lambda(\mathbf{U}) = \mathbf{RC3}(\bigcup \{ \Lambda(U) \mid U \in \mathbf{U} \})$$

Properties of $\Lambda(\mathbf{U})$

- ▶ If $\mathbf{U} \sqsubseteq \mathbf{V}$, then $\Lambda(\mathbf{U}) \subseteq \Lambda(\mathbf{V})$
- ▶ If $\mathbf{U} \sqsubseteq \mathbf{V}$, then $\Lambda(\mathbf{U}) \supseteq \Lambda(\mathbf{V})$
- ▶ If $M \in \Lambda(\mathbf{U})$ and $N \in \Lambda(\mathbf{V})$, then $M N \in \Lambda(\mathbf{U} \bullet \mathbf{V})$
- ▶ If $M[N/x] \in \Lambda(F(\mathbf{U}))$ for all $\mathbf{U} \in \text{dom}(F)$ and all $N \in \Lambda(\mathbf{U})$, then $\lambda x.M \in \Lambda(\text{fun}(F))$
- ▶ If $P\theta \in \Lambda(\mathbf{U})$, then $\theta \in \Lambda(\eta)$ for some $\eta \in \text{match}_P(\mathbf{U})$

Main Claim

If $\mathbf{U} \sqsubseteq \llbracket M \rrbracket^n_\eta$ and $\theta \in \Lambda(\eta)$, then $M\theta \in \Lambda(\mathbf{U})$.

Proof by “Scott induction”, i.e. induction on n .

Proof of “ $\text{SN}(M) \Rightarrow \llbracket M \rrbracket \neq \perp$ ”

Main Claim:

If M is strongly normalising, then

1. $\llbracket M \rrbracket \neq \perp$,
2. for every linear pattern P and every $\eta \in \text{match}_P(\llbracket M \rrbracket)$ there exists a substitution θ with $\text{dom}(\theta) = \text{FV}(P)$ such that $M \rightarrow^* P\theta$ and $\eta = \llbracket \theta \rrbracket$, i.e. $\eta(x) = \llbracket \theta \rrbracket(x)$,

Proof by induction on the strong normalisability of M .

Typed systems

Polymorphic types $\sigma \mid \iota \mid \rho \rightarrow \sigma \mid \forall p . \rho$

A *typed system* is given by a rewrite system and typing axioms $f : \vec{\rho} \rightarrow \sigma$ where f is a function constant and $\vec{\rho}, \sigma$ are types.

Typing judgements

$$\Gamma \vdash \mathbf{T} : \circ \quad \Gamma \vdash \mathbf{F} : \circ \quad \Gamma \vdash 0 : \iota \quad \frac{\Gamma \vdash M : \iota}{\Gamma \vdash (\mathbf{S}, M) : \iota}$$

$$\Gamma, x : \rho \vdash x : \rho \quad \frac{\Gamma, x : \rho \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \rho \rightarrow \sigma} \quad \frac{\Gamma \vdash M : \rho \rightarrow \sigma \quad \Gamma \vdash N : \rho}{\Gamma \vdash MN : \sigma}$$

$$\frac{\Gamma \vdash M : \rho}{\Gamma \vdash M : \forall p.\rho} \text{ (} p \text{ not free in } \Gamma \text{)} \quad \frac{\Gamma \vdash M : \forall p.\rho}{\Gamma \vdash M : \rho[\sigma/p]}$$

$$\frac{\Gamma \vdash M_i : \rho_i \quad (i = 1, \dots, k)}{\Gamma \vdash f M_1 \dots M_k : \sigma} \text{ (} f : \rho_1, \dots, \rho_k \rightarrow \sigma \text{ axiom)}$$

$$\frac{\Gamma \vdash M : \rho \quad \Gamma \vdash N : \rho}{\Gamma \vdash \mathbf{if \ then } M \mathbf{ \ else } N : \circ \rightarrow \rho}$$

Denotational semantics of types

$\llbracket \rho \rrbracket_t \subseteq D_+$ (ρ type, t : type variables \rightarrow powerset of D)

$$\llbracket \rho \rrbracket_t = t(\rho).$$

$$\llbracket o \rrbracket_t = \{T, F\}.$$

$\llbracket \iota \rrbracket_t =$ the least subset of D_+ containing 0

and with d_1, \dots, d_n also $([S], [d_1, \dots, d_n])$.

$$\llbracket \rho \rightarrow \sigma \rrbracket_t = \{\text{fun}(f) \mid f(\llbracket \rho \rrbracket_{t^*}) \subseteq \llbracket \sigma \rrbracket_{t^*}\}.$$

$$\llbracket \forall \rho . \rho \rrbracket_t = \bigcap_{A \subseteq D_+} \llbracket \rho \rrbracket_{t[p:=A]}.$$

$$\llbracket \rho \rrbracket := \llbracket \rho \rrbracket_{t_\emptyset} \quad \text{where } t_\emptyset(\rho) := \emptyset \text{ for all type variables } \rho.$$

Total typed systems

A typed system is *total* if $\llbracket f \rrbracket \in \llbracket \vec{\rho} \rightarrow \sigma \rrbracket_t^*$ for every axiom $f : \vec{\rho} \rightarrow \sigma$ and every assignment t .

Semantic typing

$$\Gamma \models M : \rho \iff \forall \eta, t (\eta \in \llbracket \Gamma \rrbracket_t^* \Rightarrow \llbracket M \rrbracket \eta \in \llbracket \rho \rrbracket_t^*)$$

Semantic soundness theorem for total typed systems

$$\Gamma \vdash M : \rho \Rightarrow \Gamma \models M : \rho$$

Corollary

Every total typed system is strongly normalising.

Type correct systems

A *type correct system* is a typed system such that

1. for every axiom $f : \rho_1, \dots, \rho_k \rightarrow \sigma$, $k \geq \text{arity}(f)$,
2. for every rewrite rule $f\vec{P} \rightarrow M$ and context Γ ,
if $\Gamma \vdash P_i : \rho_i$ for all $i = 1, \dots, k$, then $\Gamma \vdash M : \sigma$.

Lemma

Type correct systems are safe, i.e. every typeable term is safe.

Corollary

In a type correct system a typeable term is strongly normalising if and only if it does not denote \perp .

Extensions of Gödels T

$$\text{R} \quad : \quad \rho, (\iota \rightarrow \rho \rightarrow \rho), \iota \rightarrow \rho$$

$$\text{BR} \quad : \quad \tau_G, \tau_H, \tau_Y, (\iota \rightarrow \rho), \iota \rightarrow \sigma$$

where

$$\tau_G = (\iota \rightarrow \rho) \rightarrow \iota \rightarrow \sigma$$

$$\tau_H = (\iota \rightarrow \rho) \rightarrow \iota \rightarrow (\rho \rightarrow \sigma) \rightarrow \sigma$$

$$\tau_Y = (\iota \rightarrow \rho) \rightarrow \mathbf{0}$$

$$\text{OR} \quad : \quad \tau_Y, (\iota \rightarrow \rho), \iota$$

where

$$\tau_Y = (\iota \rightarrow \rho) \rightarrow (\iota \rightarrow \rho \rightarrow (\iota \rightarrow \rho) \rightarrow \iota) \rightarrow \iota$$

$$\parallel \quad \rho, \rho \rightarrow \rho$$

New strong normalisation results

Theorem

The typed system comprising system T , barrecursion, open recursion and non-deterministic choice is strongly normalising.

Proof

It suffices to show that all constants are total.

For example, $\llbracket \parallel \rrbracket \bullet \mathbf{d} \bullet \mathbf{e} = \mathbf{d}++\mathbf{e}$.

Hence, clearly $\llbracket \parallel \rrbracket \in \llbracket \rho \rightarrow \rho \rightarrow \rho \rrbracket$.

Totality of R

$$\llbracket R \rrbracket \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{d}_3 =$$

$$((0 \in \mathbf{d}_3) \triangleright \mathbf{d}_1) ++ \text{concat } [\mathbf{d}_2 \bullet \mathbf{e} \bullet (\llbracket R \rrbracket \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{e}) \mid ([S], \mathbf{e}) \leftarrow \mathbf{d}_3]$$

One easily shows

$$\llbracket R \rrbracket \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{d}_3 \in \llbracket \rho \rrbracket^*$$

for all $(\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3) \in \llbracket \rho \rrbracket^* \times \llbracket \iota \rightarrow \rho \rightarrow \rho \rrbracket^* \times \llbracket \iota \rrbracket^*$, by induction on the number of occurrences of the constructor S in $\mathbf{d}_3 \in \llbracket \iota \rrbracket^*$.

Co-products

$$\frac{\Gamma \vdash M_i : \rho_i}{\Gamma \vdash (i, M) : \rho_0 + \rho_1} \quad (i = 0, 1)$$

Case_k : $(\rho_0 \rightarrow \sigma'), (\rho_1 \rightarrow \sigma'), (\rho_0 + \rho_1), \sigma_1, \dots, \sigma_k \rightarrow \sigma,$

where $\sigma' := \sigma_1 \rightarrow \dots \rightarrow \sigma_k \rightarrow \sigma.$

$$\text{Case}_0 x_0 x_1 (i, y) \quad \rightarrow \quad x_i y \quad (i = 0, 1)$$

$$\text{Case}_{k+1} x_0 x_1 (i, y) z \quad \rightarrow \quad x_i y z \quad (i = 0, 1)$$

Permutative conversions

$$\text{Case}_{k+1} x_0 x_1 u z \rightarrow \text{Case}_k (\lambda y_0. x_0 y_0 z) (\lambda y_1. x_1 y_1 z) u.$$

Recall:

$$\text{Case}_k : (\rho_0 \rightarrow \sigma'), (\rho_1 \rightarrow \sigma'), (\rho_0 + \rho_1), \sigma_1, \dots, \sigma_k \rightarrow \sigma,$$

where $\sigma' := \sigma_1 \rightarrow \dots \rightarrow \sigma_k \rightarrow \sigma$.

Theorem

The system comprising the previously considered extensions of Gödel's T plus co-products and permutative conversions is strongly normalising.

Proof: Show that $\llbracket \text{Case}_k \rrbracket$ is total, by induction on k .

ω -rule

$$\frac{\Gamma \vdash M_0 : \rho \quad \Gamma \vdash M_1 : \rho \quad \Gamma \vdash M_2 : \rho \quad \dots}{\Gamma \vdash \langle M_i \rangle_{i \in \mathbb{N}} : \iota \rightarrow \rho}$$

$$\begin{aligned} \langle M_i \rangle_{i \in \mathbb{N}} 0 &\rightarrow M_0 \\ \langle M_i \rangle_{i \in \mathbb{N}} (S, K) &\rightarrow \langle M_{i+1} \rangle_{i \in \mathbb{N}} K \end{aligned}$$

Reduction inside $\langle M_i \rangle_{i \in \mathbb{N}}$ is not allowed.

Strong normalisation for the ω -rule*Theorem*

The system comprising the previous system plus ω -rule is strongly normalising.

Proof: The ω -rule can be simulated by constants $f_{[\vec{x}, \langle M_i \rangle_{i \in \mathbb{N}}]}$, for every infinite sequence of terms M_0, M_1, M_2, \dots with free variable included in \vec{x} , and the rewrite rules

$$\begin{aligned} f_{[\vec{x}, \langle M_i \rangle_{i \in \mathbb{N}}]} \vec{x} 0 &\rightarrow M_0 \\ f_{[\vec{x}, \langle M_i \rangle_{i \in \mathbb{N}}]} \vec{x} (S, K) &\rightarrow f_{[\vec{x}, \langle M_{i+1} \rangle_{i \in \mathbb{N}}]} \vec{x} K \end{aligned}$$

It is easy to prove that each $f_{[\vec{x}, \langle M_i \rangle_{i \in \mathbb{N}}]}$ is total.

Summary

- ▶ We introduced a flexible, powerful and easy to use method for proving strong normalisation for λ -calculi with rewriting.
- ▶ In typed systems additional constants only need to be shown total.
- ▶ Can cope with non-determinism and infinitary terms (ω -rule).
- ▶ Restricted to “definitional rules”. Hence rules like $(x + y) + z \rightarrow x + (y + z)$ are excluded.
- ▶ Can our semantic method be combined with more syntax oriented methods (Blanqui, Jouannaud, . . .), that can deal with rules like the above?

Foundational aspects

- ▶ Although easy to use, our method is too heavy handed, from a foundational point of view:
 - The definition of $\Lambda(\mathbf{U})$ requires a reflection principle (as does the usual definition of Tait's strong computability predicates).
 - The definition of $\llbracket \rho \rightarrow \sigma \rrbracket$ requires quantification over domain elements (i.e. second-order objects).
 - Similarly, the interpretation of polymorphic types requires third-order quantification.
- ▶ Due to recent results by Abel (HOR'97) it is likely that the characterisation theorem can be proven elementarily (see also Valentini's elementary proof of the corresponding theorem for intersection types).



K. Gödel.

Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica* 12 (1958) 280–287.



C. Spector.

Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. In F. D. E. Dekker, editor, *Recursive Function Theory: Proc. Sympos. in Pure Math.*, 5, 1–27. AMS, 1962



D. S. Scott.

Outline of a mathematical theory of computation. In *4th Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.



W.W. Tait.

Normal form theorem for barrecursive functions of finite type. In J.E. Fenstad, ed, *Proceedings of the Second Scandinavian Logic Symp.*, 353–367. North–Holland, 1971.



H. Vogel.

Ein starker Normalisationssatz für die barrekursiven Funktionale. *Archive for Mathematical Logic*, 18:81–84, 1976.



Yuri L. Ershov.

Model C of partial continuous functionals. *Logic Colloquium 1976*. R. Gandy and M. Hyland, editors, North Holland, Amsterdam 455–467, 1977.



G. D. Plotkin.

LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.



G. Pottinger.

A type assignment for the strongly normalisable terms. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–577. Academic Press, 1980.



H. Barendregt, , M. Coppo, and M. Dezani-Ciancaglini.

A filter lambda model and the completeness of type assignment. *Journ. Symb. Logic*, 48(4):931–940, 1983.



M. Bezem.

Strong normalization of barrecursive terms without using infinite terms. *Arch. Math. Logic*, 25:175–181, 1985.



Y. Toyama.

Counterexample to termination for the direct sum of term rewriting systems. *Information Processing Letters*, 25:141–143, 1987.



S. van Bakel.

Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102:135–163, 1992.



J. van de Pol and H. Schwichtenberg.

Strict functionals for termination proofs. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 350–364. Springer Verlag, Berlin, Heidelberg, New York, 1995.



Y. Toyama, J.W. Klop, and H.P. Barendregt.

Termination for direct sums of left-linear complete term rewriting systems. *Journal of the Association for Computing Machinery*, 42(6):1275–1304, 1995.



A. Pitts.

Relational Properties of Domains. *Information and Computation*, 127(2):66–90, 1996.



S. Berardi, M. Bezem, and T. Coquand.

On the computational content of the axiom of choice. *Journal of Symbolic Logic*, 63(2):600–622, 1998.



F. Blanqui, J-P. Jouannaud, and M. Okada.

The calculus of algebraic constructions. In P. Narendran and M. Rusinowitch, editors, *Proceedings of RTA'99*, number 1631 in LNCS, pages 301–316. Springer Verlag, Berlin, Heidelberg, New York, 1999.



S. Valentini.

An elementary proof of strong normalisation for intersection types. *Archive for Mathematical Logic*, 40(7):475–488, 2001.



B.

A computational interpretation of open induction. *Lics 2004*.



B.

Strong normalization for applied lambda calculi. *Logical Methods in Computer Science*, 1(2):1–14, 2005.



T. Coquand, A. Spiwack.

Proof of strong normalisation using domain theory. *Lics 2006*.



A. Abel.

Syntactical normalization for intersection types with term rewriting rules. In *Fourth International Workshop on Higher-Order Rewriting, HOR'07, Paris, France, 25 June 2007*, 2007.



B.

A domain model characterising strong normalisation.

To appear in *Annals of Pure and Applied Logic*.