Methods of Proof Theory in Mathematics,
Bonn, 3rd - 10th of June, 2007

A domain-theoretic strong normalisation theorem

Ulrich Berger
Swansea

7th of June, 2007

Intersection types for the λ-calculus

$$\frac{M : \rho \qquad M : \sigma}{M : \rho \cap \sigma} \qquad \frac{M : \rho \cap \sigma}{M : \rho} \qquad \frac{M : \rho \cap \sigma}{M : \sigma}$$

plus rules for function types.

Theorem (Pottinger 1980) For any λ-term M

$$M \text{ is typable} \quad \Leftrightarrow \quad M \text{ is strongly normalising}$$

The theorem is of limited practial use, because it only applies to pure λ-terms, and the left hand side is, in general, not easier to prove than the right hand side.

We will generalize it to λ-terms with rewrite rules and replace the left hand side by "M is defined" (in a natural model).

- Introduction
- λ-calculus with term rewriting
- Domain-theoretic semantics
- Characterising strong normalisation
- Types and totality
- Conclusion
- References

Examples

Gödel's system T

$$\begin{aligned} \mathrm{R}\,0 &\to A \\ \mathrm{R}\,(n+1) &\to G\,n\,(\mathrm{R}\,n) \end{aligned}$$

Spector's barrecursion

$$\mathrm{BR}(\alpha, n) \quad \to \quad \textbf{if } Y\,\alpha < n \textbf{ then } G(\alpha, n) \textbf{ else } H\,\alpha\,n\,(\lambda x.\mathrm{BR}(\alpha_n^x, n+1))$$

where

$$\begin{aligned} x < 0 &\to \mathrm{F} \\ 0 < (y+1) &\to \mathrm{T} \\ (x+1) < (y+1) &\to x < y \end{aligned}$$

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

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

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

Open Recursion

$$\mathrm{OR}\,\alpha \quad \to \quad Y\,\alpha\,(\lambda n, x, \beta.\textbf{if } x \prec \alpha\, n \textbf{ then } \mathrm{OR}\,(\alpha_n^{x,\beta}) \textbf{ else } 0\,)$$

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

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

Hence $\alpha_n^{x,\beta}$, with $x \prec \alpha\, n$, is the general form of an infinite sequence lexicographically below α.

Nondeterministic choice

$$\begin{aligned} x \parallel y &\to x \\ x \parallel y &\to y \end{aligned}$$

- Used by Kristiansen (CiE 2006) to characterise the nondeterministic polynomial hierarchy.
- Can destroy termination: Extending Gödel's T by

$$f\,0\,1\,x \to f\,x\,x\,x$$

still terminates, but adding further $\parallel$ yields

$$f\,0\,1\,(0 \parallel 1) \to f\,(0 \parallel 1)\,(0 \parallel 1)\,(0 \parallel 1) \to^2 f\,0\,1\,(0 \parallel 1)$$

(Toyama)

From semantics to SN?

- The proof-theoretic examples are all meaningful from a semantic point of view:

  The rules *define* the constants (in a suitable model).

- Does this imply strong normalisation?

From semantics to SN!

- We will define a domain-theoretic model D such that for each term M

  M has a defined value in $D \quad \Leftrightarrow \quad M$ is strongly normalising.

- The model is "natural" - hence, in many cases it is easy to prove that a term is defined, while a direct strong normalisation proof seems virtually impossible.
- The method applies to a large class of rewrite systems.

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

- "$[\![M]\!] \neq \bot \Rightarrow \mathrm{SN}(M)$" for deterministic rewrite systems, assuming SN for the underlying type theory (B 05).
- "$[\![M]\!] \neq \bot \Rightarrow \mathrm{SN}(M)$" for deterministic rewrite systems, unconditionally, using the "intersection types as filter models" idea (Coquand, Spiwack 06).

New in this talk:

- Nondeterminism.
- Completeness: "$[\![M]\!] \neq \bot \Leftrightarrow \mathrm{SN}(M)$".
- Abstract domain theory instead of formal typing rules.

Terms

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

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

Rewrite systems

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

$$f\,\vec{P} \quad \to \quad M$$

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 $\mathrm{FV}(M) \subseteq \mathrm{FV}(\vec{P})$;
- 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

$$\begin{array}{lcl} \mathrm{R}\, A\, G\, 0 & \to & A \\ \mathrm{R}\, A\, G\, (\mathrm{S}, n) & \to & G\, n\, (\mathrm{R}\, A\, G\, n) \end{array}$$

R	constant of arity 3
$0,\ \mathrm{S}$	constructors
$A,\ G,\ n$	variables

Reduction, $K \longrightarrow K'$

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

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

Strong normalisation

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

$$M \to M' \to M'' \to \ldots$$

Fullness

We only consider *full terms*, that is, terms where every constant f occurs only in contexts of the form

$$f\, M_1\, \ldots\, M_k$$

where k is the arity of f. The fullness condition also applies to the right hand sides of rules.

Fullness avoids anomalies like, for example, the following:

$$f\, x \quad \rightarrow \quad f\, x$$

The term f is strongly normalising (it is in normal form), even though any reasonable model should send f to $\perp$.

A strict reflexive Scott-domain

$$D = \mathcal{C}_\perp \oplus (D^* \times_{\mathrm{s}} D^*) \oplus (D^* \to_\perp D^*)$$

$\mathcal{C}_\perp$	flat domain of constructors
D^*	strict finite lists (non-deterministic values)
$\times_{\mathrm{s}}$	strict product
$\oplus$	strict (or coalesced) sum
$\to_\perp$	strict function space

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

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

Some important operations

$$
\begin{array}{lcll}
\mathbf{d} \bullet \mathbf{e} & = & \mathrm{concat}[f(\mathbf{e}) \mid \mathrm{fun}(f) \leftarrow \mathbf{d}] & (\mathbf{d}, \mathbf{e} \in D^*, \text{ strict}) \\
\mathrm{T} \rhd \mathbf{d} & = & \mathbf{d} & (\mathbf{d} \in D^*) \\
\mathrm{F} \rhd \mathbf{d} & = & [] & (\mathbf{d} \in D^*, \text{ non-strict!})
\end{array}
$$

$$\mathrm{match}_P \colon D^* \to (\mathrm{FV}(P) \to D^*)^*$$

$$
\begin{array}{lcl}
\mathrm{match}_x(\mathbf{d}) & = & [[x \mapsto \mathbf{d}]] \\
\mathrm{match}_c(\mathbf{d}) & = & (c \in \mathbf{d}) \rhd [\emptyset] \\
\mathrm{match}_{(P,Q)}(\mathbf{d}) & = & [\eta \cup \eta' \mid (\mathbf{e}, \mathbf{e}') \leftarrow \mathbf{d}, \eta \leftarrow \mathrm{match}_P(\mathbf{e}), \\
 & & \qquad \eta' \leftarrow \mathrm{match}_Q(\mathbf{e}')]
\end{array}
$$

The value of a term: $[\![M]\!]\eta \in D^*$

$$
\begin{array}{lcl}
[\![x]\!]\eta & = & \eta(x) \\
[\![c]\!]\_ & = & [c] \\
[\![(M,N)]\!]\eta & = & [([\![M]\!]\eta, [\![N]\!]\eta)] \\
[\![MN]\!]\eta & = & [\![M]\!]\eta \bullet [\![N]\!]\eta \\
[\![\lambda x.M]\!]\eta & = & [\mathrm{fun}(\lambda \mathbf{d} \in D^*.[\![M]\!]\eta[x := \mathbf{d}])] \\
[\![\mathbf{if}(M,N)]\!]\eta & = & [\mathrm{fun}(\lambda \mathbf{d} \in D^*.(\mathrm{T} \in \mathbf{d} \rhd [\![M]\!]\eta) +\!\!+ (\mathrm{F} \in \mathbf{d} \rhd [\![N]\!]\eta))] \\
[\![f]\!]\_ & = & [\mathrm{fun}^k(\lambda \vec{\mathbf{d}} \in (D^*)^k. \\
 & & \quad \mathrm{concat}[\,[\![M]\!]\eta \mid (\vec{P} \mapsto M) \leftarrow \mathcal{R}_f,\ \eta \leftarrow \mathrm{match}_{\vec{P}}(\vec{\mathbf{d}})\,])]
\end{array}
$$

where $\eta\colon \mathrm{FV}(M) \to D^*$ and $k = \mathrm{arity}(f)$.

The analogy with intersection types

The relation

$$\mathbf{U} \sqsubseteq [\![M]\!]\eta,$$

where $\mathbf{U}$ is a non-deterministic defined compact, 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, "$[\![M]\!] \neq \bot$", which is equivalent to "$\exists \mathbf{U}\,(\mathbf{U} \sqsubseteq [\![M]\!])$", can be read as "$M$ is typeable".

Main Theorem

For a closed term M,

$$[\![M]\!] \neq \bot \quad \Leftrightarrow \quad M \text{ is strongly normalising}$$

We sketch the proof of "$\Rightarrow$" (which doesn't need the fullness assumption).

Reducibility candidates

A term is *simple* if it is neither a constructor nor a pair nor a λ-abstraction nor an **if**-term nor of the form $f\,\vec{N}_1 \ldots N_k$ where $k < \mathrm{arity}(f)$.

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

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

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

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

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

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

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

Rank

$D = \lim_n D_n$, with canonical embeddings $\epsilon_n \colon D_n \to D$, where

$$\begin{array}{lcl} D_0 & = & \{\bot\} \\ D_{n+1} & = & \mathcal{C}_\bot \oplus (D_n^* \times_{\mathrm{s}} D_n^*) \oplus (D_n^* \to_\bot D_n^*) \end{array}$$

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

$$\begin{array}{lcl} \mathrm{rk}(U) & = & \min\{n \mid n \in \epsilon_n(D_n)\} \\ \mathrm{rk}(\mathbf{U}) & = & \sup\{\mathrm{rk}(U) \mid U \in \mathbf{U}\} \end{array}$$

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

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

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

Assigning reducibility candidates to defined compacts

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

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

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

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

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

- If $\mathbf{U} \subseteq \mathbf{V}$, 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 $\mathbf{U} \in \mathrm{dom}(F)$ and all $N \in \Lambda(\mathbf{U})$, then $\lambda x.M \in \Lambda(\mathrm{fun}(F))$
- If $P\theta \in \Lambda(\mathbf{U})$, then $\theta \in \Lambda(\eta)$ for some $\eta \in \mathrm{match}_P(\mathbf{U})$

Main Claim

If $\mathbf{U} \sqsubseteq [\![M]\!]^n \eta$ and $\theta \in \Lambda(\eta)$, then $M\theta \in \Lambda(\mathbf{U})$.

Proof by "Scott induction", i.e. induction on n.

Proof of "$\mathrm{SN}(M) \Rightarrow [\![M]\!] \neq \bot$"

Set $\eta_0(x) := []$ for all variables x.

Main Claim:

If M is strongly normalising, then

1. $[\![M]\!]\eta_0 \neq \bot$,
2. for every linear pattern P and every $\eta \in \mathrm{match}_P([\![M]\!]\eta_0)$ there exists a substitution θ with $\mathrm{dom}(\theta) = \mathrm{FV}(P)$ such that $M \to^* P\theta$ and $\eta = [\![\theta]\!]\eta_0$, i.e. $\eta(x) = [\![\theta]\!]\eta_0(x)$,

Simply typed systems

Simple types $\quad o \mid \iota \mid \rho \to \sigma$

A *simply typed system* is given by a rewrite system and a typing relation $f : \rho$ between function constants and types.

Typing judgements

$$\Gamma \vdash \mathrm{T} : o \qquad \Gamma \vdash \mathrm{F} : o \qquad \Gamma \vdash 0 : \iota \qquad \frac{\Gamma \vdash M : \iota}{\Gamma \vdash (\mathrm{S}, M) : \iota}$$

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

$$\frac{f : \rho}{\Gamma \vdash f : \rho} \qquad \frac{\Gamma \vdash M : \rho \quad \Gamma \vdash N : \rho}{\Gamma \vdash \mathbf{if}(M, N) : o \to \rho}$$

Denotational semantics of types

$$[\![\rho]\!] \subseteq D_+$$

$[\![o]\!] = \{\mathrm{T}, \mathrm{F}\}$.

$[\![\iota]\!] =$ the least subset of D_+ containing 0

and with $d_1, \ldots, d_n$ also $([\mathrm{S}], [d_1, \ldots, d_n])$.

$[\![\rho \to \sigma]\!] = \{\mathrm{fun}(f) \mid f([\![\rho]\!]^*) \subseteq [\![\sigma]\!]^*\}$.

Total simply typed systems

A simply typed system is *total* if $[\![f]\!] \in [\![\rho]\!]^*$ for every $f : \rho$.

Semantic typing $\Gamma \models M : \rho \;:\Leftrightarrow \forall \eta \in [\![\Gamma]\!]^* \; (\; [\![M]\!]\eta \in [\![\rho]\!]^* \;)$

Semantic soundness theorem for total typed systems

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

Corollary
Every total simply typed system is strongly normalising.

Applications

$$\mathrm{R} \quad : \quad \rho \to (\iota \to \rho \to \rho) \to \iota \to \rho$$

$$\mathrm{BR} \quad : \quad \tau_G \to \tau_H \to \tau_Y \to (\iota \to \rho) \to \iota \to \sigma$$

where

$$\tau_G = (\iota \to \rho) \to \iota \to \sigma$$
$$\tau_H = (\iota \to \rho) \to \iota \to (\rho \to \sigma) \to \sigma$$
$$\tau_Y = (\iota \to \rho) \to o$$

$$\mathrm{OR} \quad : \quad \tau_Y \to (\iota \to \rho) \to \iota$$

where

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

$$\| \qquad \rho \to \rho \to \rho$$

Theorem
The simply 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, $[\![\|]\!] \bullet \mathbf{d} \bullet \mathbf{e} = \mathbf{d} +\!\!+ \mathbf{e}$.

Hence, clearly $[\![\|]\!] \in [\![\rho \to \rho \to \rho]\!]$.

Totality of R

$$[\![\mathrm{R}]\!] \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{d}_3 =$$

$$((0 \in \mathbf{d}_3) \rhd \mathbf{d}_1) +\!\!+ \ \mathrm{concat}\ [\mathbf{d}_2 \bullet \mathbf{e} \bullet ([\![\mathrm{R}]\!] \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{e}) \mid ([\mathrm{S}], \mathbf{e}) \leftarrow \mathbf{d}_3]$$

One easily shows

$$[\![\mathrm{R}]\!] \bullet \mathbf{d}_1 \bullet \mathbf{d}_2 \bullet \mathbf{d}_3 \in [\![\rho]\!]^*$$

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

Polymorphism

In a polymorphic system we have in addition type variables, p, and universal types, $\forall p\,.\rho$, and the typing rules

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

Types are interpreted w.r.t. an environment ξ, assigning to every type variable p a set $\xi(p) \subseteq D_+$, by adding the clauses

$$[\![p]\!]\xi = \xi(p)$$

$$[\![\forall p\,.\rho]\!]\xi = \bigcap\{[\![\rho]\!]\xi[p := A] \mid A \subseteq D_+\}.$$

The soundness theorem remains valid (note that $[\![\rho]\!]^*$ is nonempty).

Conclusion

- 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 $(x + y) + z \longrightarrow x + (y + z)$ 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 $[\![\rho \to \sigma]\!]$ requires quantification over domain elements (i.e. second-order objects).
  - Similarly, the interpretation of polymorphic types requires third-order quantification.
- Probably, the characterisation theorem can be proven elementarily, similarly to 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*.