Electronic Notes in Theoretical Computer Science 70 No. 1 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume70.html 18 pages

Strongly Normalising Cut-Elimination with Strict Intersection Types

Steffen van Bakel

Department of Computing, Imperial College,
80 Queen's Gate, London SW7 2BZ, U.K

Abstract

This paper defines reduction on derivations in the strict intersection type assignment system of [2], by generalising cut-elimination, and shows a strong normalisation result for this reduction. Using this result, new proofs are given for the approximation theorem and the characterisation of normalisability using intersection types.

Key words: intersection types, lambda calculus, termination, cut-elimination.

Introduction

Strong normalisation of cut-elimination is a well-established property in the area of logic that has been studied profoundly, as it has been in various systems that define type assignment for the Lambda Calculus. For intersection type assignment, proofs of strong normalisation of cut-elimination have at best been indirect, i.e. obtained through a mapping from the derivations into a logic, where the property has been established before. Since there is no logic to which the type-constant ω can be adequately mapped, the intersection systems studied in this way are ω-free. This paper will use the Strict Type Assignment System of [2] (which contains ω), and will present a proof for the property directly on the derivations themselves.

The second, and perhaps more suprising, result of this paper is then that all normal characterisations of (strong/ head) normalisation are consequences of the strong normalisation of cut-elimination. Many strong normalisation results in the context of types use the technique of Computability Predicates [14,10], which provides a means for proving termination of typeable terms using a predicate defined by induction on the structure of types. This technique has been widely used to study normalisation properties (or similar results), as well as head-normalisation and approximation results (see Thm. 5.3).

[1] Email: svb@doc.ic.ac.uk

This papers considers intersection types, also because, using those types, various normalisation properties can be characterised. The Intersection Type Discipline (itd) as presented in [7] (a more enhanced system was presented in [6]; for an overview of the various existing systems, see [3]), was introduced mainly to overcome the limitations of Curry's type assignment system [8,9] and has been used to characterise normalisation using types. It is an extension of Curry's system, in that term variables (and terms) are allowed to have more than one type: in a certain context M, a term-variable x can play different, even non-unifyable, roles. This slight generalisation of Curry's system causes a great change in complexity; although type assignment in Curry's system is decidable, in itd type assignment is undecidable, since it is closed for β-equality:

$$M =_\beta N \Rightarrow (B \vdash M{:}\sigma \iff B \vdash N{:}\sigma).$$

The itd is most renown for providing proofs for the following *characterisation of (head/strong) normalisation* by assignable types (where ω is a type-constant, and stands for the universal type, i.e. all terms are typeable by ω):

$$\begin{array}{rcl} M \text{ has a head normal form} & \iff & B \vdash M{:}\sigma \ \&\ \sigma \neq \omega \\ M \text{ has a normal form} & \iff & B \vdash M{:}\sigma \ \&\ \omega \text{ does not occur in } B, \sigma \\ M \text{ is strongly normalisable} & \iff & B \vdash M{:}\sigma, \text{ where } \omega \text{ is not used at all.} \end{array}$$

These properties immediately show that type assignment, even in the system that does not contain ω [2], is undecidable.

However, in the context of *weak* reduction, the approximation result is no longer obtained via a straightforward application of the same technique. Rather, as argued and shown in [4,1], to obtain this result in the context of Combinator Systems or Term Rewriting Systems, a more general solution was needed: *strong normalisation of cut-elimination.* Perhaps surprisingly, the machinery involved to prove this gives the characterisation results for typeable terms as a corollary.

In this paper, we will show these results in the context of Lambda Calculus: we will show that cut-elimination is strongly normalising, and that all characterisation results are direct consequences of it. The added complexity of intersection types implies that, unlike for ordinary systems of type assignment, there is a significant difference between derivation reduction and ordinary reduction (see the beginning of Section 2); unlike normal typed- or type assignment system, in '$\vdash$' not every term-redex occurs with types in a derivation.

As far as cut-elimination is concerned in the context of intersection types, there exists but few related results in the literature. As [12], where a strong normalisation result was proved for derivation reduction in the setting of the notion of intersection type assignment known as $\mathcal{D}$ [11], most papers consider the BCD-system [6] without the type-constant ω. Since we consider the type ω here, together with a type inclusion relation $\leq$, that strong normalisation result itself is a true special case of the results of this paper presented in

Section 5.

The *Approximation Theorem* hinted at above is a (perhaps less known) fundamental result for ITD, and is more relevant in the context of semantics. Essentially following [15,5], the set of terms can be extended by adding the term-constant $\bot$. Adding also the reduction rules $\bot N \rightarrow_{\beta\bot} \bot$, and $\lambda x.\bot \rightarrow_{\beta\bot} \bot$ to the notion of reduction gives rise to the notion of *approximate normal forms* that are in essence finite rooted segments of Böhm-trees [5], and a model for the Lambda Calculus can be obtained by interpreting a term M by the set of approximants that can be associated to it, $\mathcal{A}(M)$. The Approximation Theorem now states that there exists a very precise relation between types assignable to a term and those assignable to its approximants and is formulated as

$$B \vdash M{:}\sigma \iff \exists A \in \mathcal{A}(M)\ [B \vdash A{:}\sigma]$$

(see [13,2,3]), it is immediately clear that the set of intersection types assignable to a term can be used to define a model for the Lambda Calculus (see [6,2,3]).

The kind of intersection type assignment considered in this paper is that of [2], i.e. the *strict* intersection type assignment system, a restricted version of the BCD-system of [6], that is equally powerful in terms of typeability and expressiveness. The major feature of this restricted system, compared to the BCD-system, a restricted version of the derivation rules and the use of strict types (first introduced in [2]); notably, the strict system is *not* closed for η-reduction.

Notation

Often $B, x{:}\sigma$ will be written for the basis $\bigcap\{B, \{x{:}\sigma\}\}$, when x does not occur in B, and we will omit the brackets '{' and '}' when writing a basis explicitly. Also, in the notation of types, as usual, right-most outer-most brackets will be omitted, and, since the type constructor $\cap$ is associative and commutative, we will write $\sigma\cap\tau\cap\rho$ rather than $(\sigma\cap\tau)\cap\rho$. Moreover, we will, when appropriate, denote $\sigma_1\cap\cdots\cap\sigma_n$ by $\cap_{\underline{n}}\sigma_i$ (where $\underline{n} = \{1,\ldots,n\}$, $n \geq 0$, and $\cap_{\underline{1}}\sigma_i = \sigma_1$) and will assume, unless stated explicitly otherwise, that each σ_i is not an intersection type.

1 Strict intersection type assignment

In this section, we will present the strict intersection type assignment system as first presented in [2], which can be seen as a restricted version of the BCD-system as presented in [6]. The major feature of this restricted system is, compared to the BCD-system, is that the $\leq$ relation on types is no longer contra-variant on arrow-types, but restricted to the one induced by $\sigma\cap\tau \leq \sigma$ and taking ω to be the maximal type.

We assume the reader to be familiar with the Lambda Calculus [5].

Definition 1.1 (i) Let Φ be a countable infinite set of type-variables, ranged over by φ. $\mathcal{T}_S$, the set of *strict types*, and the set $\mathcal{T}$ of *intersection types*, both ranged over by $\sigma, \tau, \ldots$, are defined through:

$$\begin{array}{l} \mathcal{T}_S ::= \varphi \mid (\mathcal{T} \to \mathcal{T}_S), \\ \mathcal{T} ::= (\mathcal{T}_{S1} \cap \cdots \cap \mathcal{T}_{Sn}),\ n \geq 0 \end{array}$$

We will write ω for an intersection of zero strict types.

(ii) A *statement* is an expression of the form $M{:}\sigma$, with $M \in \Lambda$, a term of the Lambda Calculus, and $\sigma \in \mathcal{T}$. M is the *subject* and σ the *predicate* of $M{:}\sigma$.

(iii) A *basis* is a partial mapping from term variables to intersection types that are not ω, and is represented as a set of statements with only distinct variables as subjects.

(iv) For bases $B_1, \ldots, B_n$, the basis $\bigcap\{B_1, \ldots, B_n\}$ is defined by: $x{:}\cap_{\underline{m}}\sigma_i \in \bigcap\{B_1, \ldots, B_n\}$ if and only if $\{x{:}\sigma_1, \ldots, x{:}\sigma_m\}$ is the (non-empty) set of all statements about x that occur in $B_1 \cup \cdots \cup B_n$.

Notice that $\mathcal{T}_S$ is a proper subset of $\mathcal{T}$.

We will consider a pre-order on types which takes into account the idempotence, commutativity and associativity of the intersection type constructor, and defines ω to be the maximal element.

Definition 1.2 [Relations on types]

(i) The relation $\leq$ is defined as the least pre-order (i.e. reflexive and transitive relation) on $\mathcal{T}$ such that:

$$\begin{array}{rcl} \cap_{\underline{n}}\sigma_i & \leq & \sigma_i, \quad \text{for all } i \in \underline{n} \\ \tau \leq \sigma_i, \quad \text{for all } i \in \underline{n} & \Rightarrow & \tau \leq \cap_{\underline{n}}\sigma_i \\ \sigma \leq \tau \leq \rho & \Rightarrow & \sigma \leq \rho \end{array}$$

(ii) The equivalence relation $\sim$ on types is defined by: $\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$, and we will consider types modulo $\sim$.

(iii) We write $B \leq B'$ if and only if for every $x{:}\sigma' \in B'$ there is an $x{:}\sigma \in B$ such that $\sigma \leq \sigma'$, and $B \sim B' \iff B \leq B' \leq B$.

Notice that $\sigma \leq \sigma$, and $\sigma \leq \omega$, for all σ; $\mathcal{T}$ may be considered modulo $\sim$; then $\leq$ becomes a partial order.

The definition of the $\leq$-relation as given in [6] (apart from dealing with intersection types occurring on the right of the arrow type constructor) also contained the alternative:

$$\rho \leq \sigma \ \&\ \tau \leq \mu \ \Rightarrow\ \sigma{\to}\tau \leq \rho{\to}\mu$$

This was added mainly to obtain a notion of type assignment closed for η-reduction ($\lambda x.Mx \to_\eta M$, if x is not free in M), a feature that is not considered here.

The following property is easy to show:

Proposition 1.3 *[cf. [3]] For all $\sigma, \tau \in \mathcal{T}$, $\sigma \leq \tau$ if and only if there are $n, m \geq 0$, σ_i $(\forall i \in \underline{n})$, τ_j $(\forall j \in \underline{m})$ such that $\sigma = \cap_{\underline{n}} \sigma_i$, $\tau = \cap_{\underline{m}} \tau_i$, and, for all $j \in \underline{m}$ there exists $i \in \underline{n}$ such that $\tau_j = \sigma_i$.* ∎

Definition 1.4 (i) *Strict intersection type assignment* and *strict intersection derivations* are defined by the following natural deduction system:

$$(Ax) : \frac{}{B, x{:}\cap_{\underline{n}} \sigma_i \vdash x : \sigma_i}\ (n \geq 0, i \in \underline{n}) \quad (\rightarrow E) : \frac{B \vdash M : \sigma \rightarrow \tau \quad B \vdash N : \sigma}{B \vdash MN : \tau}$$

$$(\cap I) : \frac{B \vdash M : \sigma_1 \quad \cdots \quad B \vdash M : \sigma_n}{B \vdash M : \cap_{\underline{n}} \sigma_i}\ (n \geq 0) \qquad (\rightarrow I) : \frac{B, x{:}\sigma \vdash M : \tau}{B \vdash \lambda x.M : \sigma \rightarrow \tau}$$

(ii) We write $B \vdash M : \sigma$ if this statement is derivable using a strict intersection derivation, and write $\mathrm{D} :: B \vdash M : \sigma$ to specify that this result was obtained through the derivation D.

To illustrate that the strict system is not closed for η-reduction, notice that we can give a derivation for $\vdash \lambda xy.xy : (\sigma \rightarrow \tau) \rightarrow (\rho \cap \sigma) \rightarrow \tau$, but cannot give a derivation for $\vdash \lambda x.x : (\sigma \rightarrow \tau) \rightarrow (\rho \cap \sigma) \rightarrow \tau$.

Notice that, since ω is considered to be the empty intersection, the derivation rule

$$(\omega) : \frac{}{B \vdash M : \omega}$$

is implicit in rule $(\cap I)$.

We will use the following notation for derivations, that aims to show the structure, in linear notation, of the derivation in terms of rules applied.

Definition 1.5 (i) If a derivation D consists of an application of rule (Ax), there n, σ_i $(\forall i \in \underline{n})$ and B such that $\mathrm{D} :: B, x{:}\cap_{\underline{n}} \sigma_i \vdash x : \sigma_j$ with $j \in \underline{n}$; we then write $\mathrm{D} = \langle Ax \rangle :: B, x{:}\cap_{\underline{n}} \sigma_i \vdash x : \sigma_j$.

(ii) If a derivation D finishes with rule $(\rightarrow I)$, there are M_1, α, β such that $\mathrm{D} :: B \vdash \lambda x.M_1 : \alpha \rightarrow \beta$, and there is a sub-derivation $\mathrm{D}_1 :: B, x{:}\alpha \vdash M_1 : \beta$ in D; we then write $\mathrm{D} = \langle \mathrm{D}_1, \rightarrow I \rangle :: B \vdash \lambda x.M_1 : \alpha \rightarrow \beta$.

(iii) If a derivation D finishes with rule $(\rightarrow E)$, there are P, Q, such that $\mathrm{D} :: B \vdash PQ : \sigma$, and there are τ and sub-derivations $\mathrm{D}_1 :: B \vdash P : \tau \rightarrow \sigma$ and $\mathrm{D}_2 :: B \vdash Q : \tau$ in D; we then write $\mathrm{D} = \langle \mathrm{D}_1, \mathrm{D}_2, \rightarrow E \rangle :: B \vdash PQ : \sigma$.

(iv) If a derivation D finishes with rule $(\cap I)$, there are σ_i $(\forall i \in \underline{n})$ such that $\mathrm{D} :: B \vdash M : \cap_{\underline{n}} \sigma_i$, and, for all $i \in \underline{n}$, there exists a $\mathrm{D}_i :: B \vdash M : \sigma_i$ that is a sub-derivation of D; we then write $\mathrm{D} = \langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I \rangle :: B \vdash M : \cap_{\underline{n}} \sigma_i$.

We will often abbreviate the short-hand notation for derivations, and, e.g., write $\langle \mathrm{D}_1, \mathrm{D}_2, \rightarrow E \rangle$ instead of $\langle \mathrm{D}_1, \mathrm{D}_2, \rightarrow E \rangle :: B \vdash PQ : \sigma$.

As shown in [2], we have the following property.

Theorem 1.6 *[cf. [2]] The following rules are admissible:*

$$(\leq): \frac{B \vdash M:\sigma}{B' \vdash M:\tau}\,(B' \leq B, \sigma \leq \tau) \qquad (=_\beta): \frac{B \vdash M:\sigma}{B \vdash N:\sigma}\,(M =_\beta N)$$

2 Derivation reduction

The notion of reduction on derivations $\mathrm{D} :: B \vdash M:\sigma$ defined in this section will follow ordinary reduction, by contracting typed redexes that occur in D, i.e. redexes for sub-terms of M of the shape $(\lambda x.P)Q$, for which the following is a sub-derivation of D:

$$\langle\langle\ \mathrm{D}_1 :: B, x{:}\rho \vdash P:\tau, {\to}I\rangle :: B \vdash \lambda x.P:\rho{\to}\tau,$$
$$\mathrm{D}_2 :: B \vdash Q:\rho, {\to}E\rangle :: B \vdash (\lambda x.P)Q:\tau,$$

A derivation of this structure will be called a *redex*, or a *cut*. We will prove in Section 4 that this notion of reduction is terminating, i.e. strongly normalisable.

The effect of this reduction will be that the derivation for the redex $(\lambda x.P)Q$ will be replaced by a derivation for the contractum $P[Q/x]$; this can be regarded as a generalisation of cut-elimination, but has, because the system at hand uses intersection types, including ω, to be defined with care, since in $\mathrm{D} :: B \vdash M:\sigma$ it is possible that M contains a redex whereas D does not.

Before formally defining reduction on derivations, we will first define a notion of substitution on derivations.

Definition 2.1 Let $\mathrm{D} :: B, x{:}\sigma \vdash M:\tau$, and $\mathrm{D}_0 :: B \vdash N:\sigma$, the derivation

$$\mathrm{D}\,[\mathrm{D}_0/x{:}\sigma] :: B \vdash M[N/x]:\tau,$$

the result of substituting D_0 for $x{:}\sigma$ in D, is inductively defined by:

(i) $\mathrm{D} = \langle Ax\rangle :: y{:}\cap_{\underline{n}}\sigma_i \vdash y:\sigma_j$ with $j \in \underline{n}$. Then $\mathrm{D}\,[\mathrm{D}_0/x{:}\sigma] = \mathrm{D}$.

(ii) $\mathrm{D} = \langle Ax\rangle :: x{:}\cap_{\underline{n}}\sigma_i \vdash x:\sigma_j$ with $j \in \underline{n}$. Then

$$\mathrm{D}_0 = \langle \mathrm{D}_0^1 :: B \vdash N:\sigma_1, \ldots, \mathrm{D}_0^n :: B \vdash N:\sigma_n, \cap I\rangle :: B \vdash N:\cap_{\underline{n}}\sigma_i,$$

so, in particular, $\mathrm{D}_0^j :: B \vdash N:\sigma_j$. Then $\mathrm{D}\,[\mathrm{D}_0/x{:}\sigma] = \mathrm{D}_0^j$.

(iii) $\mathrm{D} = \langle \mathrm{D}_1 :: B, x{:}\sigma, y{:}\alpha \vdash M_1:\beta, {\to}I\rangle :: B, x{:}\sigma \vdash \lambda y.M_1:\alpha{\to}\beta$. Let

$$\mathrm{D}' = \mathrm{D}_1\,[\mathrm{D}_0/x{:}\sigma] :: B, y{:}\alpha \vdash M_1[N/x]:\beta.$$

Then $\langle \mathrm{D}_1, {\to}I\rangle\,[\mathrm{D}_0/x{:}\sigma] = \langle \mathrm{D}', {\to}I\rangle :: B \vdash (\lambda y.M_1)[N/x]:\alpha{\to}\beta\,.$

(iv) $\mathrm{D} = \langle \mathrm{D}_1 :: B, x{:}\sigma \vdash P:\rho{\to}\tau, \mathrm{D}_2 :: B, x{:}\sigma \vdash Q:\rho, {\to}E\rangle :: B, x{:}\sigma \vdash PQ:\tau$. Let

$$\mathrm{D}_1' = \mathrm{D}_1\,[\mathrm{D}_0/x{:}\sigma] :: B \vdash P[N/x]:\rho{\to}\tau, \text{ and}$$
$$\mathrm{D}_2' = \mathrm{D}_2\,[\mathrm{D}_0/x{:}\sigma] :: B \vdash Q[N/x]:\rho,$$

then $\langle \mathrm{D}_1, \mathrm{D}_2, {\to}E\rangle[\mathrm{D}_0/x{:}\sigma] = \langle \mathrm{D}_1', \mathrm{D}_2', {\to}E\rangle :: B \vdash (PQ)[N/x]:\tau\,.$

(v) $\mathrm{D} = \langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I\rangle :: B, x{:}\sigma \vdash M{:}\cap_{\underline{n}}\tau_i$. Let, for all $i \in \underline{n}$,
$$\mathrm{D}'_i = \mathrm{D}_i\,[\mathrm{D}_0/x{:}\sigma] :: B \vdash M[N/x]{:}\tau_i,$$
then $\langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I\rangle[\mathrm{D}_0/x{:}\sigma] = \langle \mathrm{D}'_1, \ldots, D'_n, \cap I\rangle :: B \vdash M[N/x]{:}\cap_{\underline{n}}\tau_i$.

Before coming to the definition of derivation-reduction, we need to define the notion of 'position of a sub-derivation in a derivation'.

Definition 2.2 Let D be a derivation, and D′ be a sub-derivation of D. The position p of D′ in D is defined by:

(i) If $\mathrm{D}' = \mathrm{D}$, then $p = \varepsilon$.

(ii) If the position of D′ in D_1 is q, and $\mathrm{D} = \langle \mathrm{D}_1, \to I\rangle$, or $\mathrm{D} = \langle \mathrm{D}_1, \mathrm{D}_2, \to E\rangle$, then $p = 1q$.

(iii) If the position of D′ in D_2 is q, and $\mathrm{D} = \langle \mathrm{D}_1, \mathrm{D}_2, \to E\rangle$, then $p = 2q$.

(iv) If the position of D′ in D_i ($i \in \underline{n}$) is q, and $\mathrm{D} = \langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I\rangle$, then $p = q$.

We can now define a notion of reduction on derivations; notice that this reduction corresponds to contracting a redex in the term involved only if that redex appears in the derivation in a sub-derivation with type different from ω.

Definition 2.3 We say that the derivation $\mathrm{D} :: B \vdash M{:}\sigma$ reduces to $\mathrm{D}' :: B \vdash M'{:}\sigma$ at position p with redex R, if and only if:

(i) $\sigma \in \mathcal{T}_S$.
  (a) $\mathrm{D} = \langle\langle \mathrm{D}_1, \to I\rangle, \mathrm{D}_2, \to E\rangle :: B \vdash (\lambda x.M)N{:}\sigma$ (a derivation of this shape is called a *redex*). Then D reduces to
  $$\mathrm{D}_1\,[\mathrm{D}_2/x{:}\rho] :: B \vdash M[N/x]{:}\sigma$$
  at position ε with redex $(\lambda x.M)N$.
  (b) If D_1 reduces to D'_1 at position p with redex R, then
  - $\mathrm{D} = \langle \mathrm{D}_1, \to I\rangle :: B \vdash \lambda x.M_1{:}\alpha{\to}\beta$ reduces at position $1p$ with redex R to $\mathrm{D}' = \langle \mathrm{D}'_1, \to I\rangle :: B \vdash \lambda x.M'_1{:}\alpha{\to}\beta$.
  - $\mathrm{D} = \langle \mathrm{D}_1, \mathrm{D}_2, \to E\rangle :: B \vdash PQ{:}\sigma$ reduces at position $1p$ with redex R to $\mathrm{D}' = \langle \mathrm{D}'_1, \mathrm{D}_2, \to E\rangle :: P'Q{:}\sigma$.
  - $\mathrm{D} = \langle \mathrm{D}_2, \mathrm{D}_1, \to E\rangle :: B \vdash PQ{:}\sigma$ reduces at position $2p$ with redex R to $\mathrm{D}' = \langle \mathrm{D}_2, \mathrm{D}'_1, \to E\rangle :: PQ'{:}\sigma$.

(ii) $\sigma = \cap_{\underline{n}}\sigma_i$. If $\mathrm{D} :: B \vdash M{:}\cap_{\underline{n}}\sigma_i$, then, for every $i \in \underline{n}$, there are D_i, such that $\mathrm{D}_i :: B \vdash M{:}\sigma_i$, and $\mathrm{D} = \langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I\rangle$. If there is an $i \in \underline{n}$ such that D_i reduces to D'_i at position p with redex R, then, for all $j \neq i \in \underline{n}$, either
  (a) there is no redex at position p because there is no sub-derivation at that position. Let $\mathrm{R} \to_\beta \mathrm{R'}$ and $\mathrm{D}'_j = \mathrm{D}_j[\mathrm{R'}/\mathrm{R}]$ (i.e. D_j where each R is replaced by R′), or
  (b) D_j reduces to D'_j at position p with redex R.
  Then $\mathrm{D} \to_{\mathcal{D}} \langle \mathrm{D}'_1, \ldots, \mathrm{D}'_n, \cap I\rangle$ at position p with redex R.

$\mathrm{D}_1 \to_{\mathcal{D}} \mathrm{D}_2 \to_{\mathcal{D}} \mathrm{D}_3$, then $\mathrm{D}_1 \to_{\mathcal{D}} \mathrm{D}_3$.

We say, as usual, that D is *normalisable* is there exists a redex-free D′ such that D $\rightarrow_{\mathcal{D}}$ D′, and that D is *strongly normalisable* if all reduction sequences starting in D are of finite length. We abbreviate 'D is strongly normalisable' by 'SN(D)'.

It is worth noting that, when we do not allow the use of ω to type a redex, typeable terms need not be strongly normalizing, as clearly illustrated by the following example.

Example 2.4 Let D_1 be the derivation (with $B_1 = \{x{:}(\alpha{\rightarrow}\beta{\rightarrow}\gamma){\cap}\alpha, y{:}(\gamma{\rightarrow}\delta){\cap}\beta\}$, and $\Theta \equiv \lambda xy.y(xxy)$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B_1 \vdash x{:}\alpha{\rightarrow}\beta{\rightarrow}\gamma} \quad \dfrac{}{B_1 \vdash x{:}\alpha}}{B_1 \vdash xx{:}\beta{\rightarrow}\gamma} \quad \dfrac{}{B_1 \vdash y{:}\beta}}{B_1 \vdash xxy{:}\gamma}}{B_1 \vdash y{:}\gamma{\rightarrow}\delta} \quad \dfrac{}{B_1 \vdash y(xxy){:}\delta}}{B_1 \backslash y \vdash \lambda y.y(xxy){:}(\gamma{\rightarrow}\delta){\cap}\beta \rightarrow \delta}}{\vdash \Theta{:}(\alpha{\rightarrow}\beta{\rightarrow}\gamma){\cap}\alpha \rightarrow (\gamma{\rightarrow}\delta){\cap}\beta \rightarrow \delta}$$

then we can construct (with $B = \{y{:}\tau, y{:}\omega{\rightarrow}\rho\}$, and $\tau = (\alpha{\rightarrow}\beta{\rightarrow}\gamma){\cap}\alpha \rightarrow (\gamma{\rightarrow}\delta){\cap}\beta \rightarrow \delta$) D_2:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{B_2 \vdash y{:}\omega{\rightarrow}\rho} \quad \dfrac{}{B_2 \vdash xxy{:}\omega}}{B_2 \vdash y(xxy){:}\rho}}{x{:}\tau \vdash \lambda y.y(xxy){:}(\omega{\rightarrow}\rho){\rightarrow}\rho}}{\vdash \Theta{:}\tau{\rightarrow}(\omega{\rightarrow}\rho){\rightarrow}\rho}\,(\rightarrow I)$$

From D_1 and D_2 we can now construct:

$$\dfrac{\dfrac{\boxed{\mathrm{D}_2}}{\vdash \Theta{:}\tau{\rightarrow}(\omega{\rightarrow}\rho){\rightarrow}\rho} \quad \dfrac{\boxed{\mathrm{D}_2}}{\vdash \Theta{:}\tau}}{\vdash \Theta\Theta{:}(\omega{\rightarrow}\rho){\rightarrow}\rho}$$

Notice that the term $\Theta\Theta$ has only *one* redex, that is not typed with ω. This derivation has only one (derivation)-redex, and contracting it gives:

$$\dfrac{\dfrac{\dfrac{}{y{:}\omega{\rightarrow}\rho \vdash y{:}\omega{\rightarrow}\rho} \quad \dfrac{}{y{:}\omega{\rightarrow}\rho \vdash \Theta\Theta y{:}\omega}}{y{:}\omega{\rightarrow}\rho \vdash y(\Theta\Theta y){:}\rho}}{\vdash \lambda y.y(\Theta\Theta y){:}(\omega{\rightarrow}\rho){\rightarrow}\rho}$$

Notice that this last derivation is in normal form, although $\lambda y.y(\Theta\Theta y)$ obviously is not.

The following lemma formulates the relation between derivation reduction and β-reduction.

Lemma 2.5 *Let* $\mathrm{D} :: B \vdash M : \sigma$, *and* $\mathrm{D} \rightarrow_{\mathcal{D}} \mathrm{D}' :: B \vdash N : \sigma$, *then* $M \twoheadrightarrow_{\beta} N$.
Proof: By the above definition. ∎

3 Approximation

In Sections 5 and 6 we will show two main results, that are both direct consequences of the strong normalisation result proved in Section 4. Both results have been proven in the past, at least partially, in [2,3]. In fact, some of the theorems and lemmas presented here were already presented in those papers and are repeated here, for completeness, with their proofs.

The notion of approximant for lambda terms was first presented in [15], and is defined using the notion of terms in $\lambda\perp$-normal form (like in [5], $\perp$ is used, instead of Ω; also, the symbol $\sqsubseteq$ is used as a relation on $\Lambda\perp$-terms, inspired by a similar relation defined on Böhm-trees in [5]).

Definition 3.1 (i) The set of $\Lambda\perp$ *-terms* is defined as the set Λ of lambda terms, by:

$$M ::= x \mid \perp \mid \lambda x.M \mid M_1 M_2$$

The symbol $\perp$ is called *bottom*.

(ii) The notion of reduction $\rightarrow_{\beta\perp}$ is defined as $\rightarrow_{\beta}$, extended by: $\lambda x.\perp \rightarrow_{\beta\perp} \perp$ and $\perp M \rightarrow_{\beta\perp} \perp$.

(iii) The set of *normal forms for elements of* $\Lambda\perp$ *with respect to* $\rightarrow_{\beta\perp}$ is the set $\mathcal{N}$ of $\lambda\perp$*-normal forms* or *approximate normal forms*, ranged over by A, inductively defined by:

$$A ::= \perp \mid \lambda x.A \ (A \neq \perp) \mid xA_1 \cdots A_n \ (n \geq 0)$$

The rules of the system '$\vdash$' are generalised to terms containing $\perp$ by allowing for the terms to be elements of $\Lambda\perp$. Notice that, if $\perp$ occurs in a term M and $\mathrm{D} :: B \vdash M : \sigma$, then in D, $\perp$ appears in a position where the rule $(\cap I)$ is used with $n = 0$, i.e., in a sub-term typed with ω. Moreover, the terms $\lambda x.\perp$ and $\perp M_1 \cdots M_n$ are typeable by ω only.

Definition 3.2 (i) The partial order $\sqsubseteq \subseteq (\Lambda\perp)^2$ is defined as the transitive and reflexive closure of:

$$\begin{array}{rcl} \perp & \sqsubseteq & M \\ M \sqsubseteq M' & \Rightarrow & \lambda x.M \sqsubseteq \lambda x.M' \\ M_1 \sqsubseteq M_1' \ \& \ M_2 \sqsubseteq M_2' & \Rightarrow & M_1 M_2 \sqsubseteq M_1' M_2'. \end{array}$$

If $A \in \mathcal{N}$, $M \in \Lambda$, and $A \sqsubseteq M$, then A is called a *direct approximant* of M.

(ii) The relation $\underset{\sim}{\sqsubseteq} \subseteq \mathcal{N} \times \Lambda$ is defined by:

$$A \underset{\sim}{\sqsubseteq} M \iff \exists M' =_{\beta} M \, [A \sqsubseteq M'].$$

(iii) If $A \underset{\sim}{\sqsubseteq} M$, then A is called an *approximant* of M, and $\mathcal{A}(M) = \{ A \in \mathcal{N} \mid A \underset{\sim}{\sqsubseteq} M \}$.

Lemma 3.3 $B \vdash M{:}\sigma \;\&\; M \sqsubseteq M' \Rightarrow B \vdash M'{:}\sigma$.

Proof: By easy induction on the definition of $\sqsubseteq$. ∎

The following definition introduces an operation of join on $\Lambda\bot$-terms.

Definition 3.4 (i) On $\Lambda\bot$, the partial mapping *join*, $\sqcup : \Lambda\bot \times \Lambda\bot \to \Lambda\bot$, is defined by:

$$\begin{array}{rcl} \bot \sqcup M & \equiv & M \sqcup \bot \equiv M \\ x \sqcup x & \equiv & x \\ (\lambda x.M) \sqcup (\lambda x.N) & \equiv & \lambda x.(M \sqcup N) \\ (M_1 M_2) \sqcup (N_1 N_2) & \equiv & (M_1 \sqcup N_1)\,(M_2 \sqcup N_2) \end{array}$$

(ii) If $M \sqcup N$ is defined, then M and N are called *compatible*.

Note that $\bot$ can be defined as the empty join, i.e. if $M \equiv M_1 \sqcup \cdots \sqcup M_n$, and $n = 0$, then $M \equiv \bot$.

The last alternative in the definition of $\sqcup$ defines the join on applications in a more general way than Scott's, that would state that

$$(M_1 M_2) \sqcup (N_1 N_2) \sqsubseteq (M_1 \sqcup N_1)(M_2 \sqcup N_2),$$

since it is not always sure if a join of two arbitrary terms exists. However, we will use our more general definition only on terms that are compatible, so the conflict is only apparent.

The following lemma shows that the join acts as least upper bound of compatible terms.

Lemma 3.5 *If $M_1 \sqsubseteq M$, and $M_2 \sqsubseteq M$, then $M_1 \sqcup M_2$ is defined, and:*

$$M_1 \sqsubseteq M_1 \sqcup M_2,\ M_2 \sqsubseteq M_1 \sqcup M_2, \text{ and } M_1 \sqcup M_2 \sqsubseteq M.$$

Proof: By induction on the definition of $\sqsubseteq$. ∎

4 Strong normalisation of derivation reduction

In this subsection, we will prove a strong normalisation result for derivation reduction.

In order to prove that each derivation in '$\vdash$' is strongly normalisable with respect to $\to_{\mathcal{D}}$, a notion of computable derivations will be introduced.

Definition 4.1 The Computability Predicate $Comp(\mathrm{D})$ is defined recursively on types by:

$$\begin{array}{l} Comp(\mathrm{D} :: B \vdash M{:}\varphi) \iff SN(\mathrm{D}) \\ Comp(\mathrm{D} :: B \vdash M{:}\alpha{\to}\beta) \iff \\ \qquad \forall \mathrm{D}'\ [Comp(\mathrm{D}' :: B \vdash N{:}\alpha) \Rightarrow Comp(\langle \mathrm{D}, \mathrm{D}', {\to}E\rangle :: B \vdash MN{:}\beta)] \\ Comp(\langle \mathrm{D}_1, \ldots, \mathrm{D}_n, \cap I\rangle :: B \vdash M{:}\cap_{\underline{n}}\sigma_i) \iff \\ \qquad\qquad \forall i \in \underline{n}\ [Comp(\mathrm{D}_i :: B \vdash M{:}\sigma_i)] \end{array}$$

Notice that, as a special case for the third rule, we get
$Comp(\langle\cap I\rangle :: B \vdash \bot : \omega)$

We will prove that *Comp* satisfies the standard properties of computability predicates, being that computability implies strong normalisation, and that, for the so-called *neutral* objects, also the converse holds.

Lemma 4.2 *If* $Comp(\mathrm{D} :: B \vdash M : \sigma)$, $B' \leq B$, $\sigma \leq \sigma'$, *then* $Comp(\mathrm{D}' :: B' \vdash M : \sigma')$ *for some* D'.
Proof: Easy. ∎

Lemma 4.3 (i) $Comp(\mathrm{D} :: B \vdash M : \sigma) \Rightarrow SN(\mathrm{D})$.
(ii) $SN(\mathrm{D} :: B \vdash xM_1 \cdots M_m : \sigma) \Rightarrow Comp(\mathrm{D})$.
Proof: By simultaneous induction on the structure of types. ∎

The following theorem (4.5) shows that, in a derivation, replacing subderivations for term-variables by computable derivations yields a computable derivation. Before coming to this result, first an auxiliary lemmas has to be proved, that formulates that the computability predicate is closed for subject-expansion with respect to derivation reduction.

Lemma 4.4 *If* $Comp(\langle \cdots \mathrm{D}_1[\mathrm{D}_2/y{:}\rho] \cdots, \rightarrow E\rangle :: B \vdash M[Q/y]\overrightarrow{P} : \tau)$ *and* $Comp(\mathrm{D}_2 :: B \vdash Q : \rho)$, *then*

$$Comp(\langle \cdots \langle\langle \mathrm{D}_1, \rightarrow I\rangle, \mathrm{D}_2, \rightarrow E\rangle \cdots, \rightarrow E\rangle :: B \vdash (\lambda y.M)Q\overrightarrow{P} : \tau).$$

Proof: By induction on the structure of types, using Definition 4.1.

We now come to the Replacement Theorem, i.e. the proof that for every derivation in '$\vdash$', if the assumptions in the derivation are to be replaced by computable derivations, then the result itself will be computable. We will use an abbreviated notation, and write $[\overrightarrow{N/x}]$ for $[N_1/x_1, \ldots, N_n/x_n]$, etc.

Theorem 4.5 *Let* $B' = x_1{:}\mu_1, \ldots, x_m{:}\mu_m$, $\mathrm{D} :: B' \vdash M : \sigma$, *and, for every* $1 \leq i \leq m$, *there are* D_i *and* N_i *such that* $Comp(\mathrm{D}_i :: B \vdash N_i : \mu_i)$. *Then* $Comp(\mathrm{D}\,[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash M[\overrightarrow{N/x}] : \sigma)$.
Proof: By induction on the structure of derivations.
(Ax) : Then $M \equiv x$, $x{:}\cap_{\underline{n}}\sigma_i \in B'$, $\sigma = \sigma_i$ for some $i \in \underline{n}$, and $\mathrm{D}_i :: B \vdash N_1 : \sigma$. By Definition 2.1, $\mathrm{D}\,[\overrightarrow{\mathrm{D}/x{:}\mu}] = \mathrm{D}_i$.
$(\cap I)$: Then $\sigma = \cap_{\underline{n}}\sigma_i$, and, for all $i \in \underline{n}$, there exists $\mathrm{D}^i :: B' \vdash M : \sigma_j$ such that $\mathrm{D} = \langle \mathrm{D}^1, \ldots, \mathrm{D}^m, \cap I\rangle$. Then, by induction, for all $i \in \underline{n}$, $Comp(\mathrm{D}^i[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash M[\overrightarrow{N/x}] : \sigma_i)$, and, by Definition 4.1, $Comp(\mathrm{D}[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash M[\overrightarrow{N/x}] : \cap_{\underline{n}}\sigma_i)$.
$(\rightarrow I)$: Then $\sigma = \rho \rightarrow \tau$, $\mathrm{D} = \langle \mathrm{D}_1 :: B', y{:}\rho \vdash M' : \tau, \rightarrow I\rangle :: B' \vdash \lambda y.M' : \rho \rightarrow \tau$. Assume $Comp(\mathrm{D}' :: B \vdash Q : \rho)$, then:

$$\begin{array}{ll} \forall j \in \underline{m}\ [Comp(\mathrm{D}_j)]\ \&\ Comp(\mathrm{D}' :: B \vdash Q : \rho) & \Rightarrow (IH) \\ Comp(\mathrm{D}_1[\overrightarrow{\mathrm{D}/x{:}\mu}, \mathrm{D}'/y{:}\rho] :: B \vdash M[\overrightarrow{N/x}, Q/y] : \tau) & \Rightarrow \\ Comp(\langle\langle \mathrm{D}_1[\overrightarrow{\mathrm{D}/x{:}\mu}], \rightarrow I\rangle, \mathrm{D}', \rightarrow E\rangle :: B \vdash (\lambda y.M[\overrightarrow{N/x}])Q : \tau) & \end{array}$$

so, by Definition 4.1, $Comp(\langle \mathrm{D}_1[\overrightarrow{\mathrm{D}/x{:}\mu}], \rightarrow\!I\rangle :: B \vdash \lambda y.M[\overrightarrow{N/x}]{:}\rho\rightarrow\tau)$, so also $Comp(\langle \mathrm{D}_1, \rightarrow\!I\rangle[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash (\lambda y.M)[\overrightarrow{N/x}]{:}\rho\rightarrow\tau)$.

$(\rightarrow\!E)$: Then $M \equiv M_1M_2$, there are $\mathrm{D}_1, \mathrm{D}_2$, and τ such that $\mathrm{D} = \langle \mathrm{D}_1, \mathrm{D}_2, \rightarrow\!E\rangle$, $\mathrm{D}_1 :: B' \vdash M_1{:}\tau\rightarrow\sigma$, and $\mathrm{D}_2 :: B' \vdash M_2{:}\tau$. Then, by induction,

$$Comp(\mathrm{D}_1[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash M_1[\overrightarrow{N/x}]{:}\tau\rightarrow\sigma), \text{ and}$$
$$Comp(\mathrm{D}_2[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash M_2[\overrightarrow{N/x}]{:}\tau).$$

Then, by Definition 4.1,

$$Comp(\langle \mathrm{D}_1[\overrightarrow{\mathrm{D}/x{:}\mu}], \mathrm{D}_2, \rightarrow\!E\rangle :: B \vdash M_1[\overrightarrow{N/x}]M_2[\overrightarrow{N/x}]{:}\sigma)$$

so also $Comp(\langle \mathrm{D}_1, \mathrm{D}_2, \rightarrow\!E\rangle[\overrightarrow{\mathrm{D}/x{:}\mu}] :: B \vdash (M_1M_2)[\overrightarrow{N/x}]{:}\sigma)$. ■

Using this last result, we now prove a strong normalisation result for derivation reduction in '$\vdash$'.

Theorem 4.6 *If* $\mathrm{D} :: B \vdash M{:}\sigma$, *then* $SN(\mathrm{D})$.

Proof: By Lemma 4.3ii, for every $x{:}\tau \in B$, $Comp(\mathrm{D}_x :: B \vdash x{:}\tau)$, so by Theorem 4.5, $Comp(\mathrm{D} :: B \vdash M{:}\sigma)$. Notice that, by Lemma 4.3i, $SN(\mathrm{D})$. ■

5 Normalisation results

In what follows below, first an approximation result will be proved, i.e. for every M, B and σ such that $B \vdash M{:}\sigma$, there exists an $A \in \mathcal{A}(M)$ such that $B \vdash A{:}\sigma$. From this, the well-known characterisation of (head-)normalisation of lambda terms using intersection types follows easily, i.e. all terms having a (head) normal form are typeable in '$\vdash$' (with a type without ω-occurrences). The second result is the well-known characterisation of strong normalisation of typeable lambda terms, i.e. all terms, typeable in '$\vdash$' without using the type-constant ω, are strongly normalisable.

First we give some auxiliary definitions and results. The first is a notion of type assignment that, essentially, assigns ω only to the term $\bot$.

Definition 5.1 $\bot$-*type assignment* and $\bot$-*derivations* are defined by the following natural deduction system (where all types displayed are strict,

except σ in the rules $(\to I)$, and $(\to E)$):

$$(Ax): \frac{}{B, x{:}\cap_{\underline{n}}\sigma_i \vdash_\perp x{:}\sigma_i}\ (n \geq 0, i \in \underline{n})$$

$$(\to E): \frac{B \vdash_\perp M{:}\sigma\to\tau \quad B \vdash_\perp N{:}\sigma}{B \vdash_\perp MN{:}\tau}$$

$$(\cap I): \frac{B \vdash_\perp M_1{:}\sigma_1 \quad \ldots \quad B \vdash_\perp M_n{:}\sigma_n}{B \vdash_\perp M_1 \sqcup \cdots \sqcup M_n{:}\cap_{\underline{n}}\sigma_i}\ (n \geq 0)$$

$$(\to I): \frac{B, x{:}\sigma \vdash_\perp M{:}\tau}{B \vdash_\perp \lambda x.M{:}\sigma\to\tau}$$

We write $B \vdash_\perp M{:}\sigma$ if this statement is derivable using a $\perp$-derivation.

Notice that, by rule $(\cap I)$, $\emptyset \vdash_\perp \perp{:}\omega$, and that this is the only way to assign ω to a term. Moreover, in that rule, the terms M_j need to be compatible (otherwise their join would not be defined).

Lemma 5.2 (i) *If* $\mathrm{D} :: B \vdash_\perp M{:}\sigma$*, then* $\mathrm{D} :: B \vdash M{:}\sigma$*.*

(ii) *If* $\mathrm{D} :: B \vdash M{:}\sigma$*, then there exists* $M' \sqsubseteq M$*, such that* $\mathrm{D} :: B \vdash_\perp M'{:}\sigma$*.*

Proof: By easy induction on the structure of derivations, using Lemma 3.5 in part ii. ∎

Notice that, since M' need not be the same as M, the second derivation in part ii is not exactly the same; however, it has the same structure in terms of applied derivation rules.

Using Theorem 4.6, as for the BCD-system and the strict system, the relation between types assignable to a lambda term and those assignable to its approximants can be formulated as follows:

Theorem 5.3 *[Approximation]* $B \vdash M{:}\sigma \iff \exists A \in \mathcal{A}(M)\ [B \vdash A{:}\sigma]$.

Proof: $\Rightarrow$) If $\mathrm{D} :: B \vdash M{:}\sigma$, then, by Theorem 4.6, $SN(\mathrm{D})$. Let $\mathrm{D}' :: B \vdash N{:}\sigma$ be a normal form of D with respect to $\to_{\mathcal{D}}$, then by Lemma 2.5, $M \twoheadrightarrow_\beta N$ and, by Lemma 5.2ii, there exists $P \sqsubseteq N$ such that $\mathrm{D}' :: B \vdash_\perp P{:}\sigma$. So, in particular, P contains no redexes (no typed redexes since D' is in normal form, and none untyped since only $\perp$ can be typed with ω), so $P \in \mathcal{N}$, and therefore $P \in \mathcal{A}(M)$.

$\Leftarrow$) Since $A \in \mathcal{A}(\mathcal{M})$, there is an M' such that $M' =_\beta M$ and $A \sqsubseteq M'$. Then, by Lemma 3.3, $B \vdash M'{:}\sigma$, and, by Theorem 1.6, also $B \vdash M{:}\sigma$. ∎

Using this result, the following becomes easy.

Theorem 5.4 *[Head-normalisation* [3]*]* $\exists B, \sigma\ [B \vdash M{:}\sigma] \iff M$ *has a head normal form.*

Proof: $\Rightarrow$) If $B \vdash M{:}\sigma$, then, by Theorem 5.3,

$$\exists A \in \mathcal{A}(M)\ [B \vdash A{:}\sigma].$$

By Definition 3.2, there exists $M' =_\beta M$ such that $A \sqsubseteq M'$. Since $\sigma \in \mathcal{T}$, $A \not\equiv \bot$, so A is either $x, \lambda x.A'$ or $xA_1 \cdots A_n$. Since M' matches A, M' is either $x, \lambda x.M_1$ or $xM_1 \cdots M_n$; so M' is in head-normal from. Then M has a head-normal form.

$\Leftarrow$) If M has a head-normal form, then there exists $M' =_\beta M$ such that M' is either x, $\lambda x.M_1$ or $xM_1 \cdots M_n$, with each $M_i \in \Lambda$.

(a) $M' \equiv x$. Take $B = x{:}\varphi$, and $\sigma = \varphi$.

(b) $M' \equiv \lambda x.M_1$. Since M_1 is in head-normal form, by induction there are B', σ' such that $B' \vdash M_1 : \sigma'$. If $x{:}\tau \in B$, take $B = B' \backslash x$, and $\sigma = \tau \to \sigma'$, otherwise $B = B'$ and $\sigma = \omega \to \sigma'$.

(c) $M' \equiv xM_1 \cdots M_n$. Take $B = x{:}\omega \to \cdots \to \omega \to \varphi$ and $\sigma = \varphi$.

Notice that, in all cases, $B \vdash M' : \sigma$. Then, by Theorem 1.6, $B \vdash M : \sigma$.

6 ω-free type assignment

In this section we revisit the strong normalisation proof, for which we first define a notion of derivability obtained from '$\vdash$' by removing the type constant ω.

Definition 6.1 (i) The set of *ω-free strict types* is inductively defined by:

$$\sigma ::= \varphi \mid ((\sigma_1 \cap \cdots \cap \sigma_n) \to \sigma),\ (n \geq 1)$$

The set $\mathcal{T}_{\not\omega}$ of *ω-free intersection types* is defined by:

$$\{\cap_{\underline{n}}\sigma_i \mid n \geq 1 \ \&\ \forall i \in \underline{n}\ [\sigma_i \text{ is an } \omega\text{-free strict type}]\}$$

(ii) The relation $\leq$ is defined in ω-free types as the least pre-order on $\mathcal{T}_{\not\omega}$ such that:

$$\begin{array}{rrcll} & \cap_{\underline{n}}\sigma_i & \leq & \sigma_i, & \text{for all } i \in \underline{n} \\ \tau \leq \sigma_i, & \text{for all } i \in \underline{n} & \Rightarrow & \tau \leq \cap_{\underline{n}}\sigma_i & n \geq 1 \\ & \sigma \leq \tau \leq \rho & \Rightarrow & \sigma \leq \rho & \end{array}$$

(iii) The equivalence relation $\sim$ on types is defined by:
$\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$, and we will work with types modulo $\sim$.

Definition 6.2 (i) *ω-free intersection type assignment* and *ω-free intersection derivations* are defined by the following natural deduction

system:

$$(Ax): \frac{}{B, x{:}\cap_{\underline{n}}\sigma_i \vdash_{\omega} x{:}\sigma_i}\ (n \geq 1, i \in \underline{n})$$

$$(\rightarrow E): \frac{B \vdash_{\omega} M{:}\sigma{\rightarrow}\tau \quad B \vdash_{\omega} N{:}\sigma}{B \vdash_{\omega} MN{:}\tau}$$

$$(\cap I): \frac{B \vdash_{\omega} M{:}\sigma_1 \quad \cdots \quad B \vdash_{\omega} M{:}\sigma_n}{B \vdash_{\omega} M{:}\cap_{\underline{n}}\sigma_i}\ (n \geq 1)$$

$$(\rightarrow I): \frac{B, x{:}\sigma \vdash_{\omega} M{:}\tau}{B \vdash_{\omega} \lambda x.M{:}\sigma{\rightarrow}\tau}$$

(ii) We write $B \vdash_{\omega} M{:}\sigma$ if this statement is derivable using a strict intersection derivation, and write $\mathrm{D} :: B \vdash_{\omega} M{:}\sigma$ to specify that this result was obtained through the derivation D.

Then the following properties hold:

Lemma 6.3 (i) $B \vdash_{\omega} x{:}\sigma \iff \exists \sigma_i\ (\forall i \in \underline{n}) \in \mathcal{T}_{\omega}\ [x{:}\cap_{\underline{n}}\sigma_i \in B\ \&\ \exists i \in \underline{n}[\sigma = \sigma_i]]$.

(ii) $B \vdash_{\omega} MN{:}\sigma\ \&\ \sigma \in \mathcal{T}_{\omega} \iff \exists \tau \in \mathcal{T}_{\omega}\ [B \vdash_{\omega} M{:}\tau{\rightarrow}\sigma\ \&\ B \vdash_{\omega} N{:}\tau]$.

(iii) $B \vdash_{\omega} \lambda x.M{:}\sigma\ \&\ \sigma \in \mathcal{T}_{\omega} \iff \exists \rho \in \mathcal{T}_{\omega}, \mu \in \mathcal{T}_{\omega}\ [\sigma = \rho{\rightarrow}\mu\ \&\ B, x{:}\rho \vdash_{\omega} M{:}\mu]$.

(iv) $B \vdash_{\omega} M{:}\sigma\ \&\ B' \leq B\ \Rightarrow\ B' \vdash_{\omega} M{:}\sigma$.

(v) *If* $\mathrm{D} :: B \vdash_{\omega} M{:}\sigma$, *then* $\mathrm{D} :: B \vdash M{:}\sigma$.

Proof: Easy. ∎

To prepare the characterisation of terms by their assignable types, first is proved that a term in $\lambda\bot$-normal form is typeable without ω, if and only if it does not contain $\bot$. This forms the basis for the result that all normalisable terms are typeable without ω.

Lemma 6.4 [3]

(i) *If* $B \vdash A{:}\sigma$, *and* B, σ *are* ω*-free, then* A *is* $\bot$*-free.*

(ii) *If* A *is* $\bot$*-free, then there are* B, *and* σ, *such that* $B \vdash_{\omega} A{:}\sigma$.

Now, as also shown in [2], it is possible to prove that the strict intersection type assignment system satisfies the main properties of the BCD-system.

Theorem 6.5 [2] $\exists B, \sigma\ [B \vdash M{:}\sigma\ \&\ B, \sigma\ \omega\text{-free}] \iff M$ *has a normal form.*

Proof: $\Rightarrow$) If $B \vdash M{:}\sigma$, then, by Theorem 5.3, there exists $A \in \mathcal{A}(M)$ such that $B \vdash M{:}\sigma$. Since B, σ are ω -free, by Lemma 6.4i, this A is $\bot$-free. By Definition 3.1 there exists $M' =_{\beta} M$ such that $A \sqsubseteq M'$. Since A contains no $\bot$, $A = M'$, so M' is a normal form, so, especially, M has a normal form.

$\Leftarrow$) If M' is the normal form of M, then it is a $\bot$-free approximate normal form. Then by Lemma 6.4ii there are B, σ such that $B \vdash_{\omega} M' : \sigma$. Then, by Theorem 1.6, $B \vdash M : \sigma$. ∎

Theorem 6.9 shows that the set of strongly normalisable terms is exactly the set of terms typeable in the intersection system without using the type constant ω. The same result was stated in [2] for the BCD-system, but the proof there was not complete. The proof of the crucial lemma as presented below (Lemma 6.8) and part ($\Leftarrow$) of the proof of Theorem 6.9 are essentially due to Betti Venneri, of the University of Florence, Italy, and goes by induction on the left-most outer-most reduction path.

The following lemma shows a subject expansion result for the ω-free system.

Lemma 6.6 *If $B \vdash_{\omega} M[N/x] : \sigma$ and $B \vdash_{\omega} N : \rho$, then $B \vdash_{\omega} (\lambda x.M)N : \sigma$.*

Proof: Standard. ∎

This result extends by induction (easily) to all contexts: if $B \vdash_{\omega} \mathrm{C}[M[N/x]] : \sigma$ and $B \vdash_{\omega} N : \rho$, then $B \vdash_{\omega} \mathrm{C}[(\lambda x.M)N] : \sigma$.

Lemma 6.6 is also essentially the proof for the statement that each strongly normalisable term can be typed in the system '$\vdash_{\omega}$', a property that we will now show.

Definition 6.7 An occurrence of a redex $\mathrm{R} = (\lambda x.P)Q$ in a term M is called the *left-most outer-most redex of M* ($lor(M)$), if:

(i) There is no redex R' in M such that $\mathrm{R}' = \mathrm{C}[\mathrm{R}]$ (*outer-most*).

(ii) There is no redex R' in M such that $M = \mathrm{C}_0[\mathrm{C}_1[\mathrm{R}']\mathrm{C}_2[\mathrm{R}]]$ (*left-most*).

$M \to_{lor} N$ is used to indicate that M reduces to N by contracting $lor(M)$.

The following lemma formulates a subject expansion result for '$\vdash_{\omega}$' with respect to left-most outer-most reduction. The proof follows a similar proof by Betti Venneri, of the University of Florence, Italy (unpublished), set in the context of the BCD-system.

Lemma 6.8 *Let $M \to_{lor} N$, $lor(M) = (\lambda x.P)Q$, $B \vdash_{\omega} N : \sigma$, and $B' \vdash_{\omega} Q : \tau$, then there exists B_1, ρ such that $B_1 \vdash_{\omega} M : \rho$.*

Proof: By induction on the structure of types, of which only the part $\sigma \in \mathcal{T}_S$ will be shown, by induction on the structure of terms; note that $M \equiv \lambda x_1 \cdots x_k.V P_1 \cdots P_n$ $(k, n \geq 0)$, where either

(i) V is a redex, so $V \equiv (\lambda y.P)Q$, so $lor(M) = V$ and $N \equiv \lambda x_1 \cdots x_k.(P[Q/y])P_1 \cdots P_n$, or

(ii) $V \equiv y$, so there is an $j \in \underline{n}$ such that $lor(M) = lor(P_j)$, and $P_j \to_{lor} P'$, and $N \equiv \lambda x_1 \cdots x_k.yP_1 \cdots P' \cdots P_n$.

In either case, we have, by Lemma 6.3, that there are $\alpha_1, \ldots, \alpha_k$, $\gamma_1, \ldots, \gamma_n$, and β such that $\sigma = \alpha_1 \to \cdots \to \alpha_k \to \beta$, $B_0 \vdash_{\omega} V' : \gamma_1 \to \cdots \to \gamma_n \to \beta$, and $B_0 \vdash_{\omega} P_i : \gamma_i$ $(i \in \underline{n})$, where $B_0 = B, x_1{:}\alpha_1, \ldots, x_k{:}\alpha_k$, and V' is either $P[Q/y]$

or y. We distinguish two cases:
(i) $V' \equiv P[Q/y]$. Let $B_1 = B'$, then
$\bigcap\{B_0, B_1\} \vdash_{\scriptscriptstyle\omega} (\lambda y.P)Q : \gamma_1 \to \cdots \to \gamma_n \to \beta$, by Lemma 6.6,
(ii) $V' \equiv y$. Then, by induction, there are B', ρ such that $B' \vdash_{\scriptscriptstyle\omega} P_j : \rho$. Take $\mu = \gamma_1 \to \cdots \rho \cdots \to \gamma_n \to \beta, B_1 = B', y{:}\mu$, then $\bigcap\{B_0, B_1\} \vdash_{\scriptscriptstyle\omega} y : \mu$.
In either case, $\bigcap\{B_0, B_1\} \vdash_{\scriptscriptstyle\omega} V P_1 \cdots P_n : \beta$. Let, for all $i \in \underline{k}$, $x_i{:}\beta_i \in \bigcap\{B_0, B_1\}$, then

$$\bigcap\{B_0, B_1\} \backslash x_1, \ldots, x_k \vdash_{\scriptscriptstyle\omega} \lambda x_1 \cdots x_k.yP_1 \cdots P_n : \beta_1 \to \cdots \to \beta_k \to \beta. \qquad \blacksquare$$

We can now show that all strongly normalisable terms are exactly those typeable in '$\vdash_{\scriptscriptstyle\omega}$'.

Theorem 6.9 $\exists B, \sigma\ [B \vdash_{\scriptscriptstyle\omega} M : \sigma] \iff M$ *is strongly normalisable with respect to* $\to_\beta$.

Proof: $\Rightarrow$) If D :: $B \vdash_{\scriptscriptstyle\omega} M : \sigma$, then by Lemma 6.3v, also D :: $B \vdash M : \sigma$. Then, by Theorem 4.6, D is strongly normalisable with respect to $\to_{\mathcal{D}}$. Since D contains no ω, all redexes in M correspond to redexes in D. Since derivation reduction does not introduce ω, also M is strongly normalisable with respect to $\to_\beta$.
$\Leftarrow$) With induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by $\#(M)$).
(a) If $\#(M) = 0$, then M is in normal form, and by Lemma 6.4ii, there exist B and $\sigma \in \mathcal{T}$ such that $B \vdash_{\scriptscriptstyle\omega} M : \sigma$.
(b) If $\#(M) \geq 1$, so M contains a redex, then let $M \to_{lor} N$ by contracting $(\lambda x.P)Q$. Then $\#(N) \leq \#(M)$, and $\#(Q) \leq \#(M)$ (since Q is a proper sub-term of a redex in M), so by induction $B \vdash_{\scriptscriptstyle\omega} M : \sigma$ and $B' \vdash_{\scriptscriptstyle\omega} Q : \tau$, for some B, B', σ, and τ. Then, by Lemma 6.8, there exist B_1, ρ such that $B_1 \vdash_{\scriptscriptstyle\omega} M : \rho$. $\blacksquare$

Conclusions and future work

We have shown that cut-elimination is strongly normalising also for an intersection type assignment systems that contains ω, and that all standard characterisations of normalisation are consequences of this result. A future extension of this result could be to consider a type-inclusion relation that is contra-variant over the arrow, so to consider a system that is closed for η-reduction.

Acknowledgments

I am greatly indebted to Maribel Fernández for the close cooperation and never failing persistence that was needed for the development of the technique I applied here, and would like to thank Betty Venneri for allowing me to publish her proof.

References

[1] M. Fernández Bakel. Normalisation, Approximation, and Semantics for Combinator Systems. *Theoretical Computer Science*, 2002. To appear.

[2] S. Bakel. Complete restrictions of the Intersection Type Discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

[3] S. Bakel. Intersection Type Assignment Systems. *Theoretical Computer Science*, 151(2):385–435, 1995.

[4] S. Bakel and M. Fernández. Approximation and Normalization Results for Typeable Term Rewriting Systems. In Gilles Dowek, Jan Heering, Karl Meinke, and Bernhard Möller, editors, *Proceedings of HOA '95. Second International Workshop on Higher Order Algebra, Logic and Term Rewriting,* Paderborn, Germany. *Selected Papers*, volume 1074 of *Lecture Notes in Computer Science*, pages 17–36. Springer-Verlag, 1996.

[5] H. Barendregt. *The Lambda Calculus: its Syntax and Semantics.* North-Holland, Amsterdam, revised edition, 1984.

[6] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[7] M. Coppo and M. Dezani-Ciancaglini. An Extension of the Basic Functionality Theory for the λ-Calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.

[8] H.B. Curry. Functionality in Combinatory Logic. In *Proc. Nat. Acad. Sci. U.S.A.*, volume 20, pages 584–590, 1934.

[9] H.B. Curry and R. Feys. *Combinatory Logic*, volume 1. North-Holland, Amsterdam, 1958.

[10] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types.* Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[11] J-L. Krivine. *Lambda-Calcul – Types et Modèles.* Etudes et Recherches en Informatique. Masson, Paris, 1990.

[12] C. Retoré. A note on intersection types. INRIA Rapport de recherche 2431, INRIA, France, 1994.

[13] S. Ronchi Della Rocca and B. Venneri. Principal type schemes for an extended type theory. *Theoretical Computer Science*, 28:151–169, 1984.

[14] W.W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32(2):198–223, 1967.

[15] C.P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus. *SIAM J. Comput.*, 5:488–521, 1976.