Intersection types for explicit substitutions

Stéphane Lengrand[1], Pierre Lescanne[1], Dan Dougherty[2],
Mariangiola Dezani-Ciancaglini[3], and Steffen van Bakel[4]

[1] École Normale Supérieure de Lyon
46, Allée d'Italie, 69364 Lyon 07, FRANCE
E-mail: {Stephane.Lengrand,Pierre.Lescanne}@ens-lyon.fr
[2] Department of Computer Science, Worcester Polytechnic Institute
Worcester, MA 101609 USA
E-mail: dd@cs.wpi.edu
[3] Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy,
E-mail: dezani@di.unito.it,
[4] Department of Computing, Imperial College of Science, Technology and Medicine,
180 Queen's Gate, London SW7 2BZ, U.K.,
E-mail: svb@doc.ic.ac.uk,

April 15, 2003

Abstract

We present a new system of intersection types for a composition-free calculus of explicit substitutions with a rule for garbage collection, and show that it characterizes those terms which are strongly normalizing. This system extends previous work on the natural generalization of the classical intersection types system, which characterized head normalization and weak normalization, but was not complete for strong normalization. An important role is played by the notion of *available* variable in a term, which is a generalization of the classical notion of free variable.

1 Introduction

An explicit substitutions calculus is a refinement of traditional λ-calculus in which substitution is not treated as a meta-operation on terms but rather as an operation of the calculus itself. The inspiration for such a study is the observation that, in the presence of variable-binding, substitution is a complex operation to define and to implement, so that making substitutions explicit leads to a more pertinent analysis of the correctness and efficiency of compilers, theorem provers, and proof-checkers. Abadi, Cardelli, Curien, and L´evy [1] and de Bruijn [12] defined the first calculi of explicit substitutions.

Intersection type disciplines originated in [10, 11] to overcome the limitations of Curry's type assignment system and to provide a characterization of the strongly normalizing terms of the λ-calculus [33]. Since then, intersection types disciplines were used in a series of papers for characterizing evaluation properties of λ-terms [26, 25, 37, 38, 20, 2, 19, 14].

As discussed in [17] one can see an explicit substitution calculus as an improvement on both the system of combinators and the classical λ-calculus, since it is a system whose mechanics are first-order and as simple as those of combinatory logic, yet which retains the same intensional character as the traditional λ-calculus. Observe that the classical λ-calculus is a subsystem of explicit substitution systems, defined by the strategy of "eagerly" applying the substitution induced by contracting a β-redex. In this sense, explicit substitutions calculi are logically prior to classical λ-calculus, and the study of explicit substitutions represents a deeper examination of the relationship between abstraction and application.

A fundamental property of classical typed lambda-calculi is strong normalization: no term admits an infinite reduction sequence. Melliès [30] made the somewhat surprising discovery that strong normalization fails even for simply-typed terms of the Abadi-Cardelli-Curien-L´evy calculus.

Given the central place that strong normalization occupies in the theory and application of classical lambda calculus, it is important to study this property in systems of explicit substitutions. Melli´es' result exploits the existence of a *composition* operator on substitutions, and so there are two obvious and complementary directions for research. The first is to define classes of reduction strategies in the original calculus which support strong normalization; a notable example of work in this area is that of Eike Ritter [34]. The second direction is to investigate calculi in which substitutions are explicit but composition is absent; the current paper is part of this effort.

Composition-free calculi of explicit substitutions have been studied in [28, 8, 23, 7, 5] among others. Here we work in the composition-free calculus $\lambda\mathtt{x}$ [8] and a calculus $\lambda\mathtt{x}_{gc}$ obtained by adding explicit garbage collection to $\lambda\mathtt{x}$. In fact, our rule for garbage collection is stronger than the one originally presented in [8].

Previous work [16, 17] explored some reduction properties of this system using intersection types. The natural generalizations of the classical type systems were able to characterize the sets of normalizing and head-normalizing terms in terms of typability. But it was shown in [16] that the naive generalization of the classical system did not characterize the strongly normalizing terms. Typable terms were strongly normalizing but the converse fails.

Example 1.1 Let Δ be the term $\lambda u.uu$. Consider the terms

$$\begin{array}{lcl} M_1 & \equiv & ((\lambda y.z)(xx))\,\langle x=\Delta\rangle \text{ and} \\ M_2 & \equiv & z\,\langle y=xx\rangle\,\langle x=\Delta\rangle \end{array}$$

and notice that $M_1 \longrightarrow M_2$. The term M_2 is readily seen to be strongly normalizing. But M_2 is not typable in the system $\mathcal{D}$ of [16]: it is obtained from the (not strongly normalizing, hence untypable) term M_1 by contracting a β-redex, and such a contraction does not change the typing behaviour of terms under $\mathcal{D}$. Finding a type system characterizing the strongly normalizing terms was left as an open problem in [16].

Main results. In this paper we solve the aforementioned problem: we define an extension $\mathcal{E}$ of system $\mathcal{D}$ which types precisely the strongly normalizing terms. Furthermore, when a universal type ω is added, the resulting system $\mathcal{E}_\omega$ satisfies the same theorems as those in [16] characterizing the weakly normalizing, head normalizing, and solvable terms. Our claim, then, is that the system presented here — with or without a universal type — is a robust type system appropriate for analyzing reduction properties in explicit substitutions calculi.

In fact, we present two different characterizations of strong normalization, in the form of two different type systems. These systems were discovered independently [39, 27]. Each system starts with the natural generalization of the classical intersection types system to the explicit substitutions calculus and adds a new typing rule. In one system [39] the new rule essentially takes into account that, by putting a term of the shape $M\,\langle x=N\rangle$ – where x does not occur free in M – in an arbitrary context, the free variables of N will never be replaced. Therefore, we can discharge the assumptions used to type N when we derive a type for $M\,\langle x=N\rangle$. For the second system the key insight for the solution is a new notion, that of *available* variable occurrence in a term (Definition 2.2). This is a refinement of the notion of free variable.

The present paper considers both rules and gives for the so obtained system complete proofs, while [39, 27] lack some proofs.

As a corollary of our proof methods we are able to define a somewhat more general notion of garbage collection than has been studied in the literature of $\lambda\mathtt{x}$ and show that adding a reduction for garbage collection does not change the set of strongly normalizing terms.

Explicit substitutions calculi without composition typically enjoy the *preservation of strong normalization* property: a pure term is strongly normalizing in the presence of explicit substitutions if it is so under β-reduction [29, 5, 8, 6, 7, 35, 15]. It follows that the classical intersection types system does

characterize strong normalization for pure terms. In contrast, the current results provide information about *all* terms. Perhaps more significant is the fact that the proofs here are direct, involving reasoning in the explicit substitutions calculus itself, not passing through the indirection of an argument about β-reduction. Herbelin [22] has proposed also a direct proof of strong normalization for a simply typed calculus of explicit substitution which interprets sequent calculus (he restricts attention to simple types and so does not achieve a characterization of strong normalization). We recommend his introduction for other arguments on how explicit substitutions give account of the *cut rule* [18].

Recently we discovered that Jean Goubault-Larrecq [21] proposes in the exercises of his course a type system with intersection types for (a version with De Bruijn indices of) the calculus of explicit substitutions $\lambda\upsilon$ introduced in [28]. Each typable term in this calculus is shown to be strongly normalizing, but the vice versa is not true.

Plan of the paper. Section 2 presents the syntax and reduction semantics of $\lambda\mathtt{x}$, and in Section 3 we derive some important technical results about reduction, including the definition of a perpetual strategy and an inductive definition of the set of strongly normalizing terms. In Section 4 we present the type system $\mathcal{E}$ and we show the inter-derivability of the two new typing rules we define. In Section 5 we prove that all strongly normalizing terms are typable in system $\mathcal{E}$, and in Section 6 we show the converse. Finally in Section 7 we verify that the results of [16] extend to system $\mathcal{E}_\omega$.

Notation. Our notation is consistent with that of [4], to which we refer the reader for background on the classical λ-calculus. We will use $\underline{n}$ for $\{1, \ldots, n\}$.

2 The calculus $\lambda\mathtt{x}$

2.1 Syntax and available variables

Definition 2.1 The set $\Lambda\mathtt{x}$ of terms with explicit substitutions is defined as follows :

$$M, N \quad ::= \quad x \mid \lambda x.M \mid MN \mid M\langle x=N\rangle$$

A term of the form $M\langle x=N\rangle$ is called a *closure*. A term which contains no closure is called a *pure term*.

In writing terms, we will use the standard conventions for removing brackets, and use the following abbreviations:

$$\begin{array}{rcl} \overrightarrow{M} & = & M_1, \ldots, M_n \ (n \geq 0) \\ M\overrightarrow{M} & = & M M_1 \ldots M_n \ (n \geq 0) \\ M\overrightarrow{\langle x=N\rangle} & = & M\langle x_1=N_1\rangle \ldots \langle x_n=N_n\rangle \ (n \geq 0) \end{array}$$

We will see in Table 2 another description of the set of terms with explicit substitutions called *head-form taxonomy* whereas the above description could be called the *natural taxonomy*. One defines the notions of free and bound variable occurrences in a term as usual. But it turns out that in the presence of explicit substitutions a refinement of the notion of free variable, called *available* variable occurrence, is key.

Definition 2.2 The *free* variables in a term are:

$$\begin{array}{lcl} \mathit{fv}(x) & = & \{x\} \\ \mathit{fv}(\lambda x.M) & = & \mathit{fv}(M) \setminus \{x\} \\ \mathit{fv}(MN) & = & \mathit{fv}(M) \cup \mathit{fv}(N) \\ \mathit{fv}(M\langle x=N\rangle) & = & (\mathit{fv}(M)\setminus\{x\}) \cup \mathit{fv}(N) \end{array}$$

The *available* variables in a term are:

$$\begin{array}{lcl} av(x) & = & \{x\} \\ av(\lambda x.M) & = & av(M) \setminus \{x\} \\ av(MN) & = & av(M) \cup av(N) \\ av(M\langle x{=}N\rangle) & = & \begin{cases} (av(M)\setminus\{x\}) \cup av(N), & \text{if } x \in av(M) \\ av(M), & \text{if } x \notin av(M) \end{cases} \end{array}$$

A variable occurrence which is not free is called a *bound* occurrence.

For pure terms the notions of freeness and availability coincide. But availability differs from freeness in that the available variables of $M\langle x{=}N\rangle$, where x is not available in M, are exactly those of M, whereas the free variables in any case are those of M and N. The intuition is that x is not available just when the term N disappears in the course of fully applying the substitutions in $M\langle x{=}N\rangle$.

Further discussion of the motivation for defining available variable occurrences will be given after we present our type system. For now we can observe, referring to Example 1.1, that in the term $z\langle y{=}xx\rangle$ the variable x is free, but not available.

It is easy to show by induction on the structure of terms that the available variable occurrences in a term are a subset of the free variable occurrences.

Lemma 2.3 $av(M) \subseteq fv(M)$.

Notice that, actually, the calculus includes two binders, namely λ in $\lambda x.M$ which binds x in M and also $\cdot\langle\cdot{=}\cdot\rangle$ in $M\langle x{=}N\rangle$ which binds x in M. In what follows we consider terms up to a α-conversion. Throughout this paper, we will assume the Barendregt convention on variables [3] to be fulfilled: *no variable occurs both free and bound.* Since available variables are free it follows that we assume that no variable occurs both available and bound in the same context. The Barendregt convention extends to judgments $\Gamma \vdash M{:}\sigma$ (see Definition 4.5) in which variables occurring in the judgment Γ are considered as free and cannot occur bound in the term M. Thus a judgment like $(x{:}\sigma) \vdash M\langle x{=}N\rangle{:}\tau$ is prohibited by the Barendregt convention.

2.2 The rules

Definition 2.4 ($\lambda\mathtt{x}$ AND $\lambda\mathtt{x}_{gc}$) We identify the following reduction rules on $\lambda\mathtt{x}$ terms.

$$\begin{array}{lcll} (\lambda x.M)P & \longrightarrow & M\langle x{=}P\rangle & (\mathsf{B}) \\ (MN)\langle x{=}P\rangle & \longrightarrow & M\langle x{=}P\rangle N\langle x{=}P\rangle & (\mathsf{App}) \\ (\lambda y.M)\langle x{=}P\rangle & \longrightarrow & \lambda y.(M\langle x{=}P\rangle) & (\mathsf{Abs}) \\ x\langle x{=}P\rangle & \longrightarrow & P & (\mathsf{VarI}) \\ y\langle x{=}P\rangle & \longrightarrow & y & (\mathsf{VarK}) \\ M\langle x{=}P\rangle & \longrightarrow & M, \text{ if } x \notin av(M) & (\mathsf{gc}) \end{array}$$

The Barendregt convention on variables plays a major role in the above definition, especially in rule (Abs) which otherwise would involve the capture of variables. The notion of reduction $\lambda\mathtt{x}$ is obtained by deleting rule (gc), and the notion of reduction $\lambda\mathtt{x}_{gc}$ is obtained by deleting rule (VarK). The rule (gc) is called "garbage collection", as it removes useless substitutions. Notice that here we propose a form of the (gc) rule which differs from the similar rules given in [8, 17], in that it uses availability of the variable instead of freeness. It gives more generally applicable rules.

By induction on reductions one can check that the set of available variables decreases by reducing terms: clearly it coincides with the set of free variables for all pure lambda terms.

Lemma 2.5 *i) If* $M \longrightarrow N$ *then* $av(M) \supseteq av(N)$.

ii) If $x \notin av(M)$, $M \longrightarrow N$ and N is a pure term then $x \notin fv(N)$.

In contrast with the classical λ-calculus we are considering a rewrite system with several rules, which in fact interact with each other in interesting ways. For example, there is a *critical pair* formed by the rules (B) and (App), which is responsible for much of the complexity in analyzing the theory.

Definition 2.6 $(\mathcal{SN})$ We say, as usual, that M is *in normal form* if M is redex free, and write $nf(M)$ if M is in normal from. M is *normalisable* is there exists M' in normal from such that $M \longrightarrow M'$, and M is *strongly normalisable* if all reduction sequences starting in M are of finite length. We use $\mathcal{SN}$ for the set of strongly normalizing terms under $\lambda\mathtt{x}$.

3 Generation of $\mathcal{SN}$, saturated sets, and a perpetual strategy

In this section we show some properties of the set $\mathcal{SN}$: the only property which is needed for our characterization result is that $\mathcal{SN}$ is saturated (Theorem 3.4), but we think that the perpetuality of the defined strategy is by itself interesting.

3.1 An inductive characterization of $\mathcal{SN}$

We first recall a key closure condition of $\mathcal{SN}$ proved in [17].

Lemma 3.1 The set $\mathcal{SN}$ is closed under rule:

$$(\mathsf{subs}) \quad \frac{M\langle y=L\rangle\langle x=N\langle y=L\rangle\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}{M\langle x=N\rangle\langle y=L\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}$$

$$(\mathsf{gen}\text{-}\lambda) \quad \frac{M}{\lambda x.M} \qquad (\mathsf{gen}\text{-}\mathsf{var}) \quad \frac{M_1 \ldots M_n}{x\overrightarrow{M}} \qquad (\mathsf{gen}\text{-}\mathsf{B}) \quad \frac{M\langle x=N\rangle\overrightarrow{P}}{(\lambda x.M)N\overrightarrow{P}}$$

$$(\mathsf{gen}\text{-}\mathsf{App}) \quad \frac{(U\langle x=N\rangle)(V\langle x=N\rangle)\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}{(UV)\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}} \qquad (\mathsf{gen}\text{-}\mathsf{Abs}) \quad \frac{(\lambda y.M\langle x=N\rangle)\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}{(\lambda y.M)\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}$$

$$(\mathsf{gen}\text{-}\mathsf{I}) \quad \frac{N\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}{x\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}} \qquad (\mathsf{gen}\text{-}\mathsf{K}) \quad \frac{y\overrightarrow{\langle z=Q\rangle}\overrightarrow{P} \qquad N}{y\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}}$$

Table 1: Generation of $\mathcal{SN}$

Table 1 tells us how the set of strongly normalizing terms can be generated by induction. More precisely, each rule has an upper part and a lower part. Rule $(\mathsf{gen}\text{-}\mathsf{var})$ has a number (possibly zero) of terms as upper part. Top-down it is a *generation*, in that it states that if the upper terms belong to $\mathcal{SN}$, then the lower term belongs to $\mathcal{SN}$. Bottom-up it is a *characterization* of a term with respect to some other "simpler" terms: it says that if the lower term belongs to $\mathcal{SN}$, then the upper terms belong to $\mathcal{SN}$.

Proposition 3.2 $\mathcal{SN}$ is generated *by the rules of Table 1.*

Proof: It is easy to see that for each rule in Table 1, if the lower term belong to $\mathcal{SN}$ then the upper term(s) belong to $\mathcal{SN}$. The converse is more delicate.
We only consider two of the rules: $(\mathsf{gen}\text{-}\mathsf{I})$, because it is typical, and $(\mathsf{gen}\text{-}\mathsf{App})$, because it uses a specificity of this set of rules.

(gen-I) : Suppose $N\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is in $\mathcal{SN}$. Let us reason by contradiction. Suppose $x\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is not in $\mathcal{SN}$, then there is an infinite reduction starting from this term. Either this reduction never contracts the left-outermost redex $x\langle x=N\rangle$ and there exists an infinite reduction starting from N or one of the Q_i's or one of the P_j's, then $N\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is not in $\mathcal{SN}$, which is a contradiction. Or this reduction is of the form

$$\begin{array}{rcl} x\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P} & \twoheadrightarrow & x\langle x=N'\rangle\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'} \\ & \longrightarrow & N'\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'} \\ & \twoheadrightarrow & \ldots \end{array}$$

which is a contradiction with the fact that $N\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}\in\mathcal{SN}$.

(gen-App) : Suppose $(UV)\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is not in $\mathcal{SN}$, then there exists an infinite reduction. If the reduction never reduced the redex $(UV)\langle x=N\rangle$, neither by (App), nor by (B) (in which case U reduces to an abstraction), then there exists an infinite reduction starting from U, or V, or N or one of the Q_i's or one of the P_j's, then $(U\langle x=N\rangle)(V\langle x=N\rangle)\,\overrightarrow{\langle z=Q\rangle}$ is not in $\mathcal{SN}$, which is a contradiction. If

$$\begin{array}{rcl} (UV)\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P} & \twoheadrightarrow & (U'V')\langle x=N'\rangle\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'} \\ & \longrightarrow & (U'\langle x=N'\rangle)(V'\langle x=N'\rangle)\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'}, \end{array}$$

then the looked for contradiction comes from the fact that $(U\langle x=N\rangle)(V\langle x=N\rangle)\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is in $\mathcal{SN}$. Suppose now that

$$\begin{array}{rcl} (UV)\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P} & \twoheadrightarrow & ((\lambda y.U')V')\langle x=N'\rangle\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'} \\ & \longrightarrow & (U'\langle y=V'\rangle\,\langle x=N'\rangle)\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'}. \end{array}$$

But $(U\langle x=N\rangle)(V\langle x=N\rangle)\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}$ is in $\mathcal{SN}$, hence
$((\lambda y.U')\langle x=N'\rangle V'\langle x=N'\rangle)\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'}$ is in $\mathcal{SN}$, hence
$U'\langle x=N'\rangle\,\langle y=V'\langle x=N'\rangle\rangle\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'}$ is in $\mathcal{SN}$. Therefore by rule (subs) (Lemma 3.1),
$(U'\langle y=V'\rangle\,\langle x=N'\rangle)\,\overrightarrow{\langle z=Q'\rangle}\,\vec{P'}$ in $\mathcal{SN}$, which is a contradiction.

3.2 Saturated sets

In order to define the notion of saturated set we identify one new closure-condition on sets of terms.

$$\text{(gen-gc)}\quad \frac{M\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}\qquad N\in\mathcal{SN}}{M\langle x=N\rangle\,\overrightarrow{\langle z=Q\rangle}\,\vec{P}}\;(x\notin \mathit{av}(M))$$

Definition 3.3 A set closed under the rules (subs), (gen-B), (gen-App), (gen-Abs), (gen-I) and (gen-gc) is said to be $\mathcal{SN}$*-saturated*.

Theorem 3.4 (SATURATION OF $\mathcal{SN}$) *The set $\mathcal{SN}$ is saturated.*

Proof: Thanks to Lemma 3.1 and Proposition 3.2 we need only show that $\mathcal{SN}$ is closed under the new rule. To show closure under rule (gen-gc) we reformulate the proof of [17] to take into account the change from $\mathit{fv}(\cdot)$ to $\mathit{av}(\cdot)$ in the definition of $\lambda\mathtt{x}_{gc}$. We define an n*-multi-context* as a term with n holes in which we can insert n terms, or simply *multi-context* if n is understood from the context. If $C[\![\cdot,\ldots,\cdot]\!]$ is an n-multi-context and $M_1,\ldots,M_n$ are terms, then the insertions of those terms in $C[\![\cdot,\ldots,\cdot]\!]$ is denoted $C[\![M_1,\ldots,M_n]\!]$, or $C[\![\overline{M_i}]\!]$ for short. We prove the following more general statement:

Let $C[[\ldots]]$ be a multi-context, and $N_1, \ldots, N_n, M_1, \ldots, M_n$ be terms, with $x \notin av(M_i)$, for $i \in \underline{n}$. If $C[[\overrightarrow{M_i}]] \in \mathcal{SN}$ and $N_i \in \mathcal{SN}$ for $i \in \underline{n}$ then $C[[\overrightarrow{M_i\,\langle x{=}N_i\rangle}]] \in \mathcal{SN}$.

We consider triples $\langle D, \boldsymbol{M}, \boldsymbol{N}\rangle$ where D is a term, $\boldsymbol{M}$ and $\boldsymbol{N}$ are multisets of terms. Let $\sqsupset^m$ be the multiset extension [13] of $\sqsupset$, the converse of the proper subterm order, and let $\longrightarrow^m$ be the multiset extension of the reduction relation $\lambda\mathtt{x}_{gc}$. The proof is by induction over the following relation: $\langle D, \boldsymbol{M}, \boldsymbol{N}\rangle \gg \langle D', \boldsymbol{M'}, \boldsymbol{N'}\rangle$ if and only if

$$\begin{array}{rcl} D & \longrightarrow & D' \text{ or} \\ D = D' \text{ and } \boldsymbol{M} & \sqsupset^m & \boldsymbol{M'}, \text{ or} \\ D = D', \boldsymbol{M} = \boldsymbol{M'}, \text{ and } \boldsymbol{N} & \longrightarrow^m & \boldsymbol{N'}. \end{array}$$

In what follows, D will be $C[[\overrightarrow{M_i}]]$ and $\longrightarrow$ will be well-founded out of D by hypothesis; $\boldsymbol{M}$ will be $\{M_1, \ldots, M_n\}$; $\boldsymbol{N}$ will be $\{N_1, \ldots, N_n\}$ and its $\lambda\mathtt{x}_{gc}$-reducts. The relation $\longrightarrow^m$ will be well-founded since multiset extension preserves well-foundedness. Therefore, $\gg$ is well-founded and a Nötherian induction on $\gg$ is possible. A remark on cases (*iv*) and (*v*) below: in these cases the term D does not change, only its representation as $C[[\ldots]]$ does. This means we insert the N_i's at "lower" positions, allowing us to perform a Nötherian induction.

Assume the induction hypothesis and that $C[[\overrightarrow{M_i}]] \in \mathcal{SN}$ and that $N_i \in \mathcal{SN}$ for $i \in \underline{n}$. Let us prove that $C[[\overrightarrow{M_i\,\langle x{=}N_i\rangle}]]$ reduces only to terms that are in $\mathcal{SN}$.

i) $C[[\overrightarrow{M_i\,\langle x{=}N_i\rangle}]] \longrightarrow C'[[\overrightarrow{M_{i_j}\,\langle x{=}N_{i_j}\rangle}]]$ (where the $i_j \in \underline{n}$), then $C[[\overrightarrow{M_i}]] \longrightarrow C'[[\overrightarrow{M_{i_j}}]]$, and by induction $C'[[\overrightarrow{M_{i_j}\,\langle x{=}N_{i_j}\rangle}]] \in \mathcal{SN}$.

ii) $M_i \longrightarrow M_i'$, works also by induction.

iii) $N_j \longrightarrow N_j'$, works also by induction. Note that this case occurs only when the N_i are in $\mathcal{SN}$.

iv) $M_i = M_i^1 M_i^2$ and $M_i\,\langle x{=}N_i\rangle \longrightarrow M_i^1\,\langle x{=}N_i\rangle\, M_i^2\,\langle x{=}N_i\rangle$. Since

$$\{M_1, \ldots, M_i, \ldots, M_n\} \quad \sqsupset^m \quad \{M_1, \ldots, M_i^1, M_i^2, \ldots, M_n\},$$

we have $C[[M_1\,\langle x{=}N_1\rangle, \ldots, (M_i^1\,\langle x{=}N_i\rangle\, M_i^2\,\langle x{=}N_i\rangle), \ldots, M_n\,\langle x{=}N_n\rangle]] \in \mathcal{SN}$ by induction.

v) $M_i = \lambda y.M_i'$ and $M_i\,\langle x{=}N_i\rangle \longrightarrow \lambda y.(M_i'\,\langle x{=}N_i\rangle)$.

$$\{M_1, \ldots, M_i, \ldots, M_n\} \quad \sqsupset^m \quad \{M_1, \ldots, M_i', \ldots, M_n\},$$

hence $C[[M_1\,\langle x{=}N_1\rangle, \ldots, \lambda y.(M_i'\,\langle x{=}N_i\rangle), \ldots, M_n\,\langle x{=}N_n\rangle]] \in \mathcal{SN}$ by induction.

vi) $M_i\,\langle x{=}N_i\rangle \longrightarrow M_i$, which is always applicable since $x \notin av(M_i)$. Since

$$\{M_1, \ldots, M_{i-1}, M_i, M_{i+1}, \ldots, M_n\} \quad \sqsupset^m \quad \{M_1, \ldots, M_{i-1}, M_{i+1}, \ldots, M_n\},$$

also $C[[M_1\,\langle x{=}N_1\rangle, \ldots, M_i, \ldots, M_n\,\langle x{=}N_n\rangle]] \in \mathcal{SN}$ by induction.

We have shown that $\mathcal{SN}$ is closed under the rule (gen-gc). This has as a consequence that $\mathcal{SN}$ is also the set of terms strongly normalizing under $\lambda\mathtt{x}_{gc}$.

3.3 A perpetual strategy

In what follows we define a perpetual strategy, which is an extension to $\lambda\mathtt{x}$ of the strategy defined in [3], p. 338. It is based on the reduction of perpetual redexes.

Definition 3.5 (PERPETUAL REDEX) For any term not in normal form, we define its *perpetual redex*.

- The perpetual redex of $\lambda x.M$ is the perpetual redex of M.

- The perpetual redex of MN is :

MN	if MN itself is a redex
the perpetual redex of M	if M is not a normal form
the perpetual redex of N	otherwise

- The perpetual redex of $M\langle x=N\rangle$ is :

the perpetual redex of N	if $M \equiv y \neq x$ and N is not a normal form
the perpetual redex of M	if M is a closure
$M\langle x=N\rangle$	otherwise

Definition 3.6 (PERPETUAL STRATEGY) The *perpetual strategy* is the strategy that reduces always the perpetual redex. It is denoted by $\rightsquigarrow$.

Table 2 gives both the perpetual strategy and a partition of terms according to the *head-form taxonomy*. The right-hand sides of rules (perp-λ) and (perp-var) give two forms of irreducible terms when *nf*(M) and $\overrightarrow{Q}$ is empty. Then together with the left-hand sides of the other rules they split the set of terms into classes that form the *head-normal form taxonomy.*

$\lambda x.M$	$\rightsquigarrow$	$\lambda x.M'$,	if $M \rightsquigarrow M'$	(perp-λ)
MN	$\rightsquigarrow$	$M'N$, if M is not an abstraction and	$M \rightsquigarrow M'$	(perp-fun)
$x\overrightarrow{P}M\overrightarrow{Q}$	$\rightsquigarrow$	$x\overrightarrow{P}M'\overrightarrow{Q}$, if *nf*$(x\overrightarrow{P})$ and	$M \rightsquigarrow M'$	(perp-var)
$(\lambda x.M)N\overrightarrow{P}$	$\rightsquigarrow$	$M\langle x=N\rangle\overrightarrow{P}$		(perp-B)
$(UV)\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$	$\rightsquigarrow$	$(U\langle x=N\rangle)(V\langle x=N\rangle)\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$		(perp-App)
$(\lambda y.M)\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$	$\rightsquigarrow$	$(\lambda y.M\langle x=N\rangle)\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$		(perp-Abs)
$x\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$	$\rightsquigarrow$	$N\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$		(perp-I)
$y\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$	$\rightsquigarrow$	$y\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$,	if *nf*(N)	(perp-K)
$y\langle x=N\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$	$\rightsquigarrow$	$y\langle x=N'\rangle\overrightarrow{\langle z=Q\rangle}\overrightarrow{P}$,	if $N \rightsquigarrow N'$	(perp-clo)

Table 2: The perpetual strategy and the head-form taxonomy

Since each term contains at most one perpetual redex, the perpetual strategy is deterministic. Note that, in the case of $\lambda\mathtt{x}_{gc}$, the perpetual strategy never reduces by (gc), except when (gc) is degenerated into (VarK), which means that in this case the perpetual redex is of the form $y\langle x=N\rangle$.

The perpetual strategy is intended to terminate on a term only when the term is strongly normalizing. This is why it does not reduce a term $y\langle x=N\rangle$ by (VarK) or (gc) when N is not a normal form. Indeed, if N is not strongly normalising, the perpetual strategy (to be really perpetual) has to reduce N instead of causing it to disappear.

Theorem 1 *The following are equivalent*

- $M \in \mathcal{SN}$.
- *The perpetual strategy terminates on* M.

Proof: For the non-trivial direction, examine the inductive characterization of $\mathcal{SN}$ and observe that when M is not strongly normalizing and has the form of the conclusion of one of the inference rules there, one of the hypotheses of the rule is obtained from M by the perpetual strategy.

4 The system $\mathcal{E}$ of intersection types

We will consider intersection types as first defined in [11] with a pre-order which takes the idempotence, commutativity and associativity of the intersection type constructor into account.

Definition 4.1 The set of *types*, ranged over by $\sigma, \tau, \rho, \ldots$, is inductively defined as follows.

$$\tau_1, \tau_2 \quad ::= \quad \varphi \mid \tau_1 \cap \tau_2 \mid \tau_1 \to \tau_2$$

where φ ranges over a denumerable set of type atoms.
The standard pre-ordering $\leq$ on types is the smallest transitive and reflexive relation such that

$$\begin{array}{rcl} \tau_1 \cap \tau_2 & \leq & \tau_1, \\ \tau_1 \cap \tau_2 & \leq & \tau_2, \\ \text{if } \sigma \leq \tau_1 \text{ and } \sigma \leq \tau_2 & \text{then} & \sigma \leq \tau_1 \cap \tau_2 \end{array}$$

The pre-order defines the equivalence relation on types :

$$\tau \sim \sigma \quad \text{if and only if} \quad \tau \leq \sigma \text{ and } \sigma \leq \tau$$

In the notation of types, as usual, $\cap$ takes precedence over $\to$, 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 $(\sigma \cap \tau) \cap \rho$.

The notion of environment is standard, but defining the union of environments requires some care for the presence of the intersection type constructor.

Definition 4.2 An *environment* is a partial assignment from variables to types, where each individual assignment is written $(x{:}\tau)$. Environments are partially ordered as follows.

$$\Gamma \leq \Gamma' \quad \text{iff} \quad (x{:}\tau') \in \Gamma' \Longrightarrow (\exists \tau).(x{:}\tau) \in \Gamma \text{ and } \tau \leq \tau'$$

By abuse of notation, we write $x \in \Gamma$ for $(\exists \tau).(x{:}\tau) \in \Gamma$. $\Gamma \backslash x$ is the environment which does not contain x in its domain and which assigns the same type as Γ to the other variables.

Notice that the direction of the ordering $\leq$ on environments may seem at first somewhat counter-intuitive: for example, in the case where for each τ and τ' we have $\tau = \tau'$, $\Gamma \leq \Gamma'$ means $\Gamma \supseteq \Gamma'$. But as we will see, $\Gamma \leq \Gamma'$ can be thought of as an extension of $\leq$ to types.

Definition 4.3
$$\begin{array}{rcl} \Gamma_1 \sqcap \Gamma_2 & = & \{(x{:}\tau) \mid (x{:}\tau) \in \Gamma_1 \ \& \ x \notin \Gamma_2\} \cup \\ & & \{(x{:}\tau) \mid (x{:}\tau) \in \Gamma_2 \ \& \ x \notin \Gamma_1\} \cup \\ & & \{(x{:}\tau_1 \cap \tau_2) \mid (x{:}\tau_1) \in \Gamma_1 \ \& \ (x{:}\tau_2) \in \Gamma_2\} \\ \Gamma, (x{:}\tau) & = & \Gamma \backslash x \cup \{(x{:}\tau)\} \end{array}$$

For example, $\{(x{:}\tau_1)\} \sqcap \{(x{:}\tau_2)\}$ denotes $\{(x{:}\tau_1 \cap \tau_2)\}$, while $\{(x{:}\tau_1)\}, (x{:}\tau_2)$ denotes $\{(x{:}\tau_2)\}$.

Lemma 4.4
- $\Gamma_1 \sqcap \Gamma_2 \leq \Gamma_1$ *and* $\Gamma_1 \sqcap \Gamma_2 \leq \Gamma_2$.
- *If* $\Gamma_1 \leq \Gamma$ *and* $\Gamma_2 \leq \Gamma$ *then* $\Gamma_1 \sqcap \Gamma_2 \leq \Gamma$.

Proof: These are routine verifications. ■

As discussed in the introduction, the key of our type assignment are non-standard cut-rules which allow to forget the context of the minor premise.

Definition 4.5 (TYPE ASSIGNMENT RULES) *Type assignments* for terms in $\Lambda\mathsf{x}$ forms the system $\mathcal{E}$ defined as follows:

$$(\mathsf{start})\quad \frac{}{\Gamma \vdash x:\sigma}\,((x{:}\sigma)\in\Gamma) \qquad (\mathsf{cut})\quad \frac{\Gamma,(x{:}\sigma)\vdash M:\tau \quad \Gamma\vdash N:\sigma}{\Gamma\vdash M\langle x=N\rangle:\tau}$$

$$(\to\mathsf{I})\quad \frac{\Gamma,(x{:}\sigma)\vdash M:\tau}{\Gamma\vdash \lambda x.M:\sigma\to\tau} \qquad (\to\mathsf{E})\quad \frac{\Gamma\vdash M:\sigma\to\tau \quad \Gamma\vdash N:\sigma}{\Gamma\vdash MN:\tau}$$

$$(\cap\mathsf{I})\quad \frac{\Gamma\vdash M:\sigma \quad \Gamma\vdash M:\tau}{\Gamma\vdash M:\sigma\cap\tau} \qquad (\cap\mathsf{E})\quad \frac{\Gamma\vdash M:\sigma_1\cap\sigma_2}{\Gamma\vdash M:\sigma_i}\,(i\in\{1,2\})$$

$$(\mathsf{drop})\quad \frac{\Gamma\vdash M:\tau \quad \Delta\vdash N:\sigma}{\Gamma\vdash M\langle x=N\rangle:\tau}\,(x\notin av(M)) \qquad (\mathsf{K\text{-}cut})\quad \frac{\Gamma\vdash M:\tau \quad \Delta\vdash N:\sigma}{\Gamma\vdash M\langle x=N\rangle:\tau}\,(x\notin\Gamma)$$

We write $\Gamma\vdash M:\sigma$ if there exists a derivation constructed using the above rules that has this statement as its conclusion.

The type system of [17] is obtained by removing the last two inference rules: the point of view taken there was that a closure $M\langle x=N\rangle$ should always have the same typing behaviour as the B-redex $(\lambda x.M)N$ which yields it. This is a plausible strategy since B-reduction involves no (immediate) erasing of sub-terms, even when x is not free in M; and indeed the resulting system — in the presence of a universal type — yields the expected characterizations of head-normalizing and left-most-normalizing terms. But as we have seen in Example 1.1 this system failed to provide a characterization of the strongly normalizing terms. This example makes clear that we must allow the type system to distinguish between certain B-redexes and their contractions.

Perhaps one's first instinct is to note that in Example 1.1 the input variable of the B-redex in M_1 does not occur free in the function body (*i.e.*, we have a "K-redex" in classical λ-calculus). This suggests modifying the cut-rule to obtain one which, when typing $M\langle x=N\rangle$ with x not free in M, relaxes the typing hypothesis for N to merely ask that it be typable under *some* environment. This seems particularly appropriate since it echoes the hypotheses of the Subject Expansion Theorem in treatments of intersection types for classical λ-calculus. But such a rule doesn't work: it is still too restrictive. For example, the reader can easily check that the term $M_2\equiv z\langle y=xx\rangle\langle x=S\rangle$ cannot be typed in such a system since $x\in fv(z\langle y=xx\rangle)$, but it is clearly strongly normalizing. This example should motivate our notion of *available* variable occurrence and the corresponding typing rule (drop).

One can also observe that no premise for x is necessary when typing z in M_2 and this leads to the introduction of rule $(\mathsf{K\text{-}cut})$.

A good exercise at this point is to check that the term M_2 can be typed in system $\mathcal{E}$. On the other hand, notice that rule (cut) has no side-condition, therefore, when $x\notin av(M)$ and $\Gamma\vdash N:\sigma$, one can freely use (cut) or (drop), and when $x\notin\Gamma$ and $\Gamma\vdash N:\sigma$, one can freely use (cut) or $(\mathsf{K\text{-}cut})$.

The following are some elementary properties of the type system, which enlighten the relations between the non-standard cut rules.

Lemma 4.6 *i) If* $\Gamma'\le\Gamma$, $\tau\le\tau'$ *and* $\Gamma\vdash M:\tau$ *then* $\Gamma'\vdash M:\tau'$.
ii) If $x\in av(M)$, *then* $\Gamma\vdash M:\tau$ *implies* $x\in\Gamma$.
iii) If $x\notin av(M)$, *then* $\Gamma\vdash M:\tau$ *implies* $\Gamma\backslash x\vdash M:\tau$.
iv) If $x\notin av(M)$, *then* $\Gamma\vdash M:\tau$ *implies* $\Gamma,(x{:}\sigma)\vdash M:\tau$ *for any type* σ.

Proof: By induction on the structure of derivations, with the exception of part (*iv*) which follows immediately from parts (*i*) and (*iii*).

- Before proving part (i) it is useful to make the following observation. Let M^y_z denote the result of substituting (in the traditional sense) y for z in M, and let Γ^y_z be the obvious extension of this notion to environments. If $\Gamma \vdash M:\tau$ then $\Gamma^y_z \vdash M^y_z:\tau$ (this follows by a straightforward induction). Now, in proving part (i) the only non-trivial case is when the last applied rule is $(\mathsf{K\text{-}cut})$:

$$(\mathsf{K\text{-}cut}) \quad \frac{\Gamma \vdash P:\tau \quad \Delta \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,(y \notin \Gamma)$$

  Now, if y did not occur in Γ', the argument would be a simple appeal to the induction hypothesis. But there is no reason to assume this, so we have to work a little. Let y' be a fresh variable, not occurring (free) in Γ', Δ, P, or N. Since $\Gamma' \leq \Gamma$ we know that y' does not occur in Γ. By our observation about the preservation of derivations under ordinary substitution, $\Gamma \vdash P^{y'}_y:\tau$. So by induction $\Gamma' \vdash P^{y'}_y:\tau'$. Thus $\Gamma' \vdash P^{y'}_y\langle y'=N\rangle:\tau'$ by $(\mathsf{K\text{-}cut})$. But $P^{y'}_y\langle y'=N\rangle$ is α-equivalent with $P\langle y=N\rangle$, so we are done.
- For part (ii) three cases have to be looked at. The first one is when M is $P\langle y=N\rangle$ and the derivation ends with

$$(\mathsf{cut}) \quad \frac{\Gamma, (y{:}\sigma) \vdash P:\tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,.$$

  Since $x \in av(M)$, by Lemma 2.3, x is free in M and by the variable convention and the fact that y is bound, we get $x \neq y$. By the definition of available variable, x available in $M \equiv P\langle y=N\rangle$ means that $x \in av(P)$ or $x \in av(N)$. In both cases the induction hypothesis yields $x \in \Gamma$. The other cases are

$$(\mathsf{drop}) \quad \frac{\Gamma \vdash P:\tau \quad \Delta \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,(y \notin av(P)) \qquad\qquad (\mathsf{K\text{-}cut}) \quad \frac{\Gamma \vdash P:\tau \quad \Delta \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,(y \notin \Gamma)$$

  In the case of $(\mathsf{K\text{-}cut})$ notice that, by induction, $y \notin \Gamma$ implies $y \notin av(P)$. So in each case, from $x \in av(M)$ we get $x \in av(P)$. We may then conclude, by induction, that $x \in \Gamma$. ■

4.1 Derivable rules

By Definition 4.5, the rules of system $\mathcal{E}$ are (start), $(\rightarrow\mathsf{I})$, $(\rightarrow\mathsf{E})$, $(\cap\mathsf{I})$, $(\cap\mathsf{E})$, (cut), (drop), and $(\mathsf{K\text{-}cut})$. $\mathcal{DLL}$ is the system obtained from $\mathcal{E}$ by dropping rule $(\mathsf{K\text{-}cut})$ and $v\mathcal{BD}$ is the systems obtained from $\mathcal{E}$ by dropping rule (drop). We write $\Gamma \vdash_{\mathcal{DLL}} M:\sigma$ if there exists a derivation with rules in $\mathcal{DLL}$ that has this as its conclusion. We write $\Gamma \vdash_{v\mathcal{BD}} M:\sigma$ if there exists a derivation with rules in $v\mathcal{BD}$ that has this as its conclusion.

We will show that these systems have the same typing power as system $\mathcal{E}$, so we can say that just one of the rules $(\mathsf{K\text{-}cut})$ and (drop) suffices.

Lemma 4.7 *i) Rule* $(\mathsf{K\text{-}cut})$ *is derivable in system* $\mathcal{DLL}$.
ii) Rule (drop) *is derivable in system* $v\mathcal{BD}$.

Proof: *i*) Each application of rule $(\mathsf{K\text{-}cut})$

$$(\mathsf{K\text{-}cut}) \quad \frac{\Gamma \vdash P:\tau \quad \Delta \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,(y \notin \Gamma)$$

can be replaced by an application of rule (drop), since, by Lemma 4.6 (ii), $y \notin \Gamma$ implies $y \notin av(P)$.

ii) Consider an application of rule drop:

$$(\mathsf{drop}) \quad \frac{\Gamma \vdash P:\tau \quad \Delta \vdash N:\sigma}{\Gamma \vdash P\langle y=N\rangle:\tau}\,(y \notin av(P))$$

By Lemma 4.6. (*iii*) $\Gamma\backslash y \vdash P:\tau$. Then the (K-cut) rule yields $\Gamma\backslash y \vdash P\langle y=N\rangle:\tau$. Then by Lemma 4.6 (*iv*) we have $\Gamma \vdash P\langle y=N\rangle:\tau$.

From the above Lemma we easily get:

Theorem 4.8 *The sets of derivable judgments in systems $\mathcal{E}$, $\mathcal{DLL}$, and $v\mathcal{BD}$ coincide.*

5 Typing strongly normalizing terms

As usual for type assignment systems, we have a Generation Lemma.

Lemma 5.1 (GENERATION LEMMA)

i) $\Gamma \vdash x:\sigma$ *if and only if there exists* $(x:\tau) \in \Gamma$ *such that* $\tau \leq \sigma$.

ii) $\Gamma \vdash MN:\sigma$ *if and only if there exist* n, *and* σ_i, τ_i $(i \in \underline{n})$ *such that* $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$, *and* $\Gamma \vdash M:\tau_i \to \sigma_i$ *and* $\Gamma \vdash N:\tau_i$.

iii) $\Gamma \vdash \lambda x.M:\sigma$ *if and only if there exist* n, *and* ρ_i, τ_i $(i \in \underline{n})$ *such that* $\sigma \sim ((\rho_1 \to \tau_1) \cap \ldots \cap (\rho_n \to \tau_n))$, *and* $\Gamma, (x:\rho_i) \vdash M:\tau_i$ *whenever* $i \in \underline{n}$.

iv) $\Gamma \vdash M\langle x=N\rangle:\sigma$ *if and only if either*

a) $x \in av(M)$, *and there exists* τ *such that* $\Gamma, (x:\tau) \vdash M:\sigma$ *and* $\Gamma \vdash N:\tau$, *or*

b) $x \notin av(M)$, $\Gamma \vdash M:\sigma$ *and* N *is typeable.*

Proof: The right-to-left implications immediately follow from the typing rules. The proof of the vice-versa by induction on derivations is easy. For part (*iv*) notice that Theorem 4.8 allows us to skip (K-cut) rule. If the last applied rule is ($\cap$I) we can use Lemma 4.6 (*i*) and rule ($\cap$I) . ■

A minimal requirement of our system is that it satisfies the subject reduction property (SR). We will show SR for the reduction $\lambda\mathtt{x}_{gc}$: this gives us SR for $\lambda\mathtt{x}$ for free.

Theorem 5.2 (SUBJECT REDUCTION) *If* $M \longrightarrow N$, *then* $\Gamma \vdash M:\tau$ *implies* $\Gamma \vdash N:\tau$.

Proof: By induction on the definition of the reduction relation, ' $\longrightarrow$ '. We only show the base cases.

(B) : Then $\Gamma \vdash (\lambda x.M)N:\sigma$, and, by Lemma 5.1 (*ii*), there exist types σ_i, ρ_i $(i \in \underline{n})$ such that $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$, and, for all $i \in \underline{n}$, $\Gamma \vdash \lambda x.M:\rho_i \to \sigma_i$ and $\Gamma \vdash N:\rho_i$. We can assume none of the σ_i to be an intersection, so, by Lemma 5.1 (*iii*) $\Gamma, (x:\rho_i) \vdash M:\sigma_i$, and therefore, by rule (cut), $\Gamma \vdash M\langle x=N\rangle:\sigma_i$. So, by rule ($\cap$I), $\Gamma \vdash M\langle x=N\rangle:\sigma$.

(App) : Then $\Gamma \vdash (MN)\langle x=P\rangle:\sigma$. Let $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$ where none of the σ_i is an intersection. By Lemma 5.1 (*iv*), we have two cases:

($x \in av(MN)$ *and there exists* τ *such that* $\Gamma, (x:\tau) \vdash MN:\sigma$ *and* $\Gamma \vdash P:\tau$) : Then $x \in av(M)$ or $x \in av(N)$, and, by Lemma 5.1 (*ii*), for every $i \in \underline{n}$, there exists ρ_i such that $\Gamma, (x:\tau) \vdash M:\rho_i \to \sigma_i$ and $\Gamma, (x:\tau) \vdash N:\rho_i$. Then, by rule (cut), $\Gamma \vdash M\langle x=P\rangle:\rho_i \to \sigma_i$ and $\Gamma \vdash N\langle x=P\rangle:\rho_i$.

($x \notin av(MN)$, $\Gamma \vdash MN:\sigma$ *and there exist* Δ, τ *such that* $\Delta \vdash P:\tau$) : Then $x \notin av(M)$ and $x \notin av(N)$. As above, by Lemma 5.1 (*ii*), there exists ρ_i such that $\Gamma \vdash M:\rho_i \to \sigma_i$ and $\Gamma \vdash N:\rho_i$. Then, by rule (drop) $\Gamma \vdash M\langle x=P\rangle:\rho_i \to \sigma_i$ and $\Gamma \vdash N\langle x=P\rangle:\rho_i$.

In both cases, by rule ($\to$E), we get $\Gamma \vdash (M\langle x=P\rangle)(N\langle x=P\rangle):\sigma_i$, so by rule ($\cap$I), $\Gamma \vdash (M\langle x=P\rangle)(N\langle x=P\rangle):\sigma$.

(Abs) : Then $\Gamma \vdash (\lambda y.M)\langle x=N\rangle : \sigma$. Let $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$ where none of the σ_i is an intersection. By Lemma 5.1 (*iv*), we have two cases:

($x \in av(M)$, *and there exists* τ *such that* $\Gamma, (x{:}\tau) \vdash \lambda y.M : \sigma$ *and* $\Gamma \vdash N : \tau$) : By Lemma 5.1 (*iii*), for $i \in \underline{n}$, there exist ρ_i, μ_i such that $\sigma_i \sim \rho_i \to \mu_i$ and $\Gamma, (x{:}\tau), (y{:}\rho_i) \vdash M : \mu_i$. Then, by rule (cut), $\Gamma, (y{:}\rho_i) \vdash M\langle x=N\rangle : \mu_i$.

($x \notin av(M)$, $\Gamma \vdash \lambda y.M : \sigma$ *and there exist* Δ, τ *such that* $\Delta \vdash N : \tau$) : As above, there exist ρ_i, μ_i such that $\sigma_i \sim \rho_i \to \mu_i$ and $\Gamma\backslash x, (y{:}\rho_i) \vdash M : \mu_i$. Then, by rule (drop), $\Gamma, (y{:}\rho_i) \vdash M\langle x=N\rangle : \mu_i$.

In both cases, by rule ($\to$I), $\Gamma \vdash \lambda y.(M\langle x=N\rangle) : \sigma_i$, and, by rule ($\cap$I), $\Gamma \vdash \lambda y.(M\langle x=N\rangle) : \sigma$.

(VarI) : Then $\Gamma \vdash x\langle x=N\rangle : \sigma$, and, by Lemma 5.1 (*iv*) there exists τ such that $\Gamma, (x{:}\tau) \vdash x : \sigma$. and $\Gamma \vdash N : \tau$. Then, by Lemma 5.1 (*i*), $\tau \leq \sigma$, and, by Lemma 4.6 (*i*) we get $\Gamma \vdash N : \sigma$.

(gc) : Then $\Gamma \vdash M\langle x=N\rangle : \sigma$ and $x \notin av(M)$. Then, by Lemma 5.1 (*iv*) we get $\Gamma \vdash M : \sigma$. ■

Normal forms in Λx are the same as in classical λ-calculus, and the type system $\mathcal{E}$ is an extension of the standard system of intersection types for classical λ-calculus. Therefore we get for free the typability of all normal forms. Moreover, we show that λ-free normal forms have arbitrary types: this also holds in the the standard system of intersection types.

Lemma 5.3 (NORMAL FORMS ARE TYPABLE) *Let M be a normal form.*

i) If M is a λ-free and τ is a type, then there is an environment in which M has type τ.

ii) M is typable.

Proof: By simultaneous structural induction on M.

- If M is a variable, both statements hold.
- If $M \equiv xM_1 \ldots M_n$ where $M_1, \ldots, M_n$ are normal forms, then by induction there are, for $i \in \underline{n}$, Γ_i, τ_i such that $\Gamma_i \vdash M_i : \tau_i$. Then $\Gamma_1 \sqcap \ldots \Gamma_n \sqcap \{x{:}\tau_1 \to \ldots \to \tau_n \to \tau\} \vdash M : \tau$. So M is typable with an arbitrary type τ in a suitable environment.
- If $M \equiv \lambda x.M'$, then by induction (second statement), there are Γ and τ such that $\Gamma \vdash M' : \tau$. Then $\Gamma, (x{:}\sigma) \vdash M' : \tau$, where either $(x{:}\sigma) \in \Gamma$ or $x \notin \Gamma$ and σ is any type. Hence, $\Gamma\backslash x \vdash M : \sigma \to \tau$. ■

The key property to obtain the typability of all strongly normalizing terms is the preservation of typability when we expand using the perpetual strategy. This comes as a corollary of the following more technical theorem.

Theorem 5.4 (SUBJECT EXPANSION) *If $M \rightsquigarrow N$ in one step, then*

i) if the rule applied in the reduction is not (B)*:* $\Gamma \vdash N : \tau \quad \Rightarrow \quad \Gamma \vdash M : \tau$

ii) if the rule applied in the reduction is (B)*:*

$$\Gamma \vdash N : \tau \quad \Rightarrow \quad \begin{cases} \Gamma \vdash M : \tau & \text{if } M \text{ is a closure} \\ (\exists\, \Gamma' \leq \Gamma).\ \Gamma' \vdash M : \tau & \text{if } M \text{ is not an abstraction} \\ (\exists\, \tau', \Gamma' \leq \Gamma).\ \Gamma' \vdash M : \tau' & \text{if } M \text{ is an abstraction} \end{cases}$$

Proof: (*i*) The proof is by structural induction on M. The base case is when M is its own perpetual redex: let us reason by cases on the rule used.

(App) : We assume $\Gamma \vdash P\langle x=U\rangle\, Q\langle x=U\rangle : \sigma$, and we want to prove $\Gamma \vdash (PQ)\langle x=U\rangle : \sigma$. By Lemma 5.1 (*ii*), there are types τ_i, σ_i ($i \in \underline{n}$) such that $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$, and

$$(\forall\, i \in \underline{n}).\Gamma \vdash P\langle x=U\rangle : \tau_i \to \sigma_i \ \&\ \Gamma \vdash Q\langle x=U\rangle : \tau_i$$

By rule ($\cap$I) it suffices to prove that $(\forall\, i \in \underline{n}).\Gamma \vdash (PQ)\langle x=U\rangle : \sigma_i$. If $x \notin av(P)$ and $x \notin av(Q)$, we apply Lemma 5.1 (*iv*), which gives $\Gamma \vdash P : \tau_i \to \sigma_i$ and $\Gamma \vdash Q : \tau_i$, as well as

that U *is typable*. Consequently, $\Gamma \vdash PQ:\sigma_i$ and finally by rule (drop) $\Gamma \vdash (PQ)\langle x{=}U\rangle:\sigma_i$. If $x \in av(P)$ or $x \in av(Q)$, it suffices to prove

$$(\exists \tau_i').\Gamma \vdash U:\tau_i' \quad \& \quad \Gamma,(x{:}\tau_i') \vdash P:\tau_i{\to}\sigma_i \quad \& \quad \Gamma,(x{:}\tau_i') \vdash Q:\tau_i$$

(which induces by rule (drop) $\Gamma \vdash (PQ)\langle x{=}U\rangle:\sigma_i$). In each case, we apply Lemma 5.1 (*iv*) on both P and Q.

- If $x \in av(P)$ and $x \notin av(Q)$, we get μ such that $\Gamma,(x{:}\mu) \vdash P:\tau_i{\to}\sigma_i$ and $\Gamma \vdash U:\mu$. Taking τ_i' to be μ, we use 4.6 (*iv*) on Q to get the result.
- If $x \notin av(P)$ and $x \in av(Q)$, we get ν such that $\Gamma,(x{:}\nu) \vdash Q:\tau_i$ and $\Gamma \vdash U:\nu$. Taking τ_i' to be ν, we use 4.6 (*iv*) on P to get the result.
- If $x \in av(P)$ and $x \in av(Q)$, we get μ and ν, such that
  - $\Gamma \vdash U:\mu$,
  - $\Gamma \vdash U:\nu$,
  - $\Gamma,(x{:}\mu) \vdash P:\tau_i{\to}\sigma_i$ and
  - $\Gamma,(x{:}\nu) \vdash Q:\tau_i$

  If we set τ_i' to $\mu{\cap}\nu$ we get the result.

(Abs) : Suppose $M \equiv (\lambda y.P)\langle x{=}U\rangle$ and $N \equiv \lambda y.(P\langle x{=}U\rangle)$. By Barendregt's convention, $y \notin av(U)$ and $x \neq y$; then $x \in av(P)$ if and only if $x \in av(\lambda y.P)$. We assume $\Gamma \vdash \lambda y.(P\langle x{=}U\rangle):\sigma$, and want to prove $\Gamma \vdash (\lambda y.P)\langle x{=}U\rangle:\sigma$. Using Lemma 5.1 (*iii*), we have types τ_i,σ_i $(i \in \underline{n})$ such that $\sigma \sim ((\tau_1{\to}\sigma_1){\cap}\ldots{\cap}(\tau_n{\to}\sigma_n))$ and $(\forall i \in \underline{n}).\Gamma,(y{:}\tau_i) \vdash P\langle x{=}U\rangle:\sigma_i$. By rule ($\cap$I) it suffices to prove that $(\forall i \in \underline{n}).\Gamma \vdash (\lambda y.P)\langle x{=}U\rangle:\tau_i{\to}\sigma_i$. We apply Lemma 5.1 (*iv*) on $\Gamma,(y{:}\tau_i) \vdash P\langle x{=}U\rangle:\sigma_i$ and thereby,

- If $x \in av(P)$ we get μ such that $\Gamma,(y{:}\tau_i),(x{:}\mu) \vdash P:\sigma_i$ and $\Gamma,(y{:}\tau_i) \vdash U:\mu$. Since $y \notin av(U)$ is assumed, applying Lemma 4.6 (*i*) will get $\Gamma,(x{:}\mu),(y{:}\tau_i) \vdash P:\sigma_i$ and $\Gamma \vdash U:\mu$, whereby we get the result.
- If $x \notin av(P)$ we get that U is *typable* and $\Gamma,(y{:}\tau_i) \vdash P:\sigma_i$ as required.

(VarI) : If $\Gamma \vdash U:\tau$, then clearly $\Gamma,(x{:}\tau) \vdash x:\tau$ and $\Gamma \vdash x\langle x{=}U\rangle:\tau$.

(VarK) : Then U is a normal form, and, by Lemma 5.3, U is *typable*, so if $\Gamma \vdash y:\sigma$, the rule (drop) yields $\Gamma \vdash y\langle x{=}U\rangle:\sigma$.

Now for the induction step, since the environment and the type of M are the same as of N, the proof is easy using the same typing tree.

(*ii*) Again, the proof is by structural induction on M.

(M *is its own perpetual redex*) : We wish to prove: if $\Gamma \vdash P\langle x{=}U\rangle:\tau$, then $(\exists \Gamma'' \leq \Gamma).\Gamma'' \vdash (\lambda x.P)U:\tau$.

- If $x \in av(P)$, we have $(\exists \tau').\Gamma,(x{:}\tau') \vdash P:\tau$ & $\Gamma \vdash U:\tau'$, so $(\exists \tau').\Gamma \vdash \lambda x.P:\tau'{\to}\tau$ & $\Gamma \vdash U:\tau'$ which entails $\Gamma \vdash (\lambda x.P)U:\tau$ by rule ($\to$E).
- If $x \notin av(P)$, then, using Lemma 5.1 (*iv*), we have $\Gamma \vdash P:\tau$ and $(\exists \Gamma',\tau').\Gamma' \vdash U:\tau'$. From Lemma 4.6. (*i*), we get $\Gamma,(x{:}\tau') \vdash P:\tau$ which yields $\Gamma \vdash \lambda x.P:\tau'{\to}\tau$ by rule ($\to$I). Hence $(\exists \Gamma',\tau').\Gamma \vdash \lambda x.P:\tau'{\to}\tau$ & $\Gamma' \vdash U:\tau'$. If we set Γ'' to be $\Gamma \sqcap \Gamma' \leq \Gamma$ we get $\Gamma'' \vdash \lambda x.P:\tau'{\to}\tau$ and $\Gamma'' \vdash U:\tau'$ which entails $\Gamma'' \vdash (\lambda x.P)U:\tau$.

($M \equiv \lambda x.M'$) : $N \equiv \lambda x.N'$, where $M' \rightsquigarrow N'$. We assume $\Gamma \vdash \lambda x.N':\sigma$ and we want to prove $\Gamma' \vdash \lambda x.M':\sigma'$ for some environment $\Gamma' \leq \Gamma$ and type σ'. Using Lemma 5.1 (*iii*), we have types τ_i,σ_i $(i \in \underline{n})$ such that $(\forall i \in \underline{n}).\Gamma,(y{:}\tau_i) \vdash N':\sigma_i$. Then, by induction, we get $\Gamma' \leq \Gamma$, τ_1', and σ_1' such that $\Gamma',(x{:}\tau_1') \vdash M':\sigma_1'$. Taking $\sigma' := \tau_1'{\to}\sigma_1'$ we get $\Gamma' \vdash \lambda x.M':\sigma'$ as required.

($M \equiv M_1M_2$ *where* M *is not its own perpetual redex*) : $N \equiv N_1N_2$ where either $M_1 \rightsquigarrow N_1$ or M_1 is a λ-free normal form and $M_2 \rightsquigarrow N_2$. We assume $\Gamma \vdash N_1N_2:\sigma$, and we want to prove

$\Gamma' \vdash M_1 M_2 : \sigma$ for some environment $\Gamma' \leq \Gamma$. Using Lemma 5.1 (*ii*), we have types τ_i, σ_i $(i \in \underline{n})$ such that $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$ and $(\forall i \in \underline{n}).\Gamma \vdash N_1 : \tau_i \to \sigma_i$ & $\Gamma \vdash N_2 : \tau_i$. Using Lemma 4.6 (*i*) it suffices to prove that $(\forall i \in \underline{n}).\Gamma_i \vdash M_1 M_2 : \sigma_i$ for some $\Gamma_i \leq \Gamma$ (since then we can take Γ' to be $(\Gamma_1 \sqcap \ldots \sqcap \Gamma_n) \leq \Gamma_i \leq \Gamma$). Now by Def. 3.5, M_1 cannot be an abstraction, otherwise M would be its own perpetual redex.

- If $M_1 \rightsquigarrow N_1$ and $M_2 \equiv N_2$, then we apply the induction hypothesis to M_1. Hence we have $\Gamma_i \leq \Gamma$ such that $\Gamma_i \vdash M_1 : \tau_i \to \sigma_i$, and using Lemma 4.6 (*i*) we get $\Gamma_i \vdash M_2 : \tau_i$. Hence $\Gamma_i \vdash M_1 M_2 : \sigma_i$.
- If $M_2 \rightsquigarrow N_2$ and $M_1 \equiv N_1$, then we apply the induction hypothesis to M_2. Hence we have $\Gamma'_i \leq \Gamma$ and τ'_i such that $\Gamma'_i \vdash M_2 : \tau'_i$. But we know by Def. 3.5 that M_1 is a normal form, which is by the way λ-free, so applying Lemma 5.3 (*i*) provides an environment Γ'' in which M_1 has type $\tau'_i \to \sigma_i$. Now by taking Γ_i to be $\Gamma'_i \sqcap \Gamma''$, we get $\Gamma_i \vdash M_1 M_2 : \sigma_i$ as required.

$(M \equiv M_1 \langle x = M_2 \rangle)$: By Def. 3.5, either:

- The perpetual redex of M is in M_2, and $M_1 \equiv y \neq x$. (Hence, $N \equiv y \langle x = N_2 \rangle$ where $M_2 \rightsquigarrow N_2$). Then assume $\Gamma \vdash y \langle x = N_2 \rangle : \sigma$. Using Lemma 5.1 (*iv*), we get $\Gamma \vdash y : \sigma$. Now by induction we get that M_2 is *typable*. Hence applying the rule (drop) we get $\Gamma \vdash y \langle x = M_2 \rangle : \sigma$ as required.
- The perpetual redex of M is in M_1, and M_1 is a closure. (Hence, $N \equiv N_1 \langle x = M_2 \rangle$ where $M_1 \rightsquigarrow N_1$). We assume $\Gamma \vdash N_1 \langle x = M_2 \rangle : \sigma$, and we want to prove $\Gamma \vdash M_1 \langle x = M_2 \rangle : \sigma$.
  - If $x \in av(N_1)$, then, using Lemma 5.1 (*iv*), we have a type τ such that $\Gamma, (x{:}\tau) \vdash N_1 : \sigma$ and $\Gamma \vdash M_2 : \tau$. Now we can apply the induction hypothesis to M_1, which is a closure. We get $\Gamma, (x{:}\tau) \vdash M_1 : \sigma$, and then we can apply rule (cut) to get $\Gamma \vdash M_1 \langle x = M_2 \rangle : \sigma$.
  - If $x \notin av(N_1)$, then using Lemma 4.6 (*iii*) we get $\Gamma \backslash x \vdash N_1 \langle x = M_2 \rangle : \sigma$. Then we can apply Lemma 5.1 (*iv*), and we have $\Gamma \backslash x \vdash N_1 : \sigma$ and M_2 is *typable*. Now we can apply the induction hypothesis to M_1, which is a closure. We get $\Gamma \backslash x \vdash M_1 : \sigma$. Note that since $x \notin (\Gamma \backslash x)$, we can apply rule (K-cut) and get $\Gamma \vdash M_1 \langle x = M_2 \rangle : \sigma$.

Corollary 5.5 (WEAK SUBJECT EXPANSION) *If $M \rightsquigarrow N$, then N is typable implies M is typable.*

Theorem 5.6 *All strongly normalizing terms are typable.*

Proof: We may induct on the length of the perpetual derivation. For the base case we observe that normal forms are typable (Lemma 5.3 (*ii*)), and the induction step is Corollary 5.5.

6 All Typeable Terms are Strongly Normalizable

The general idea of the reducibility method is to interpret types by suitable sets (saturated and stable sets for Tait [36] and Krivine [24] and admissible relations for Mitchell [31, 32]) of terms (*reducible terms*) which satisfy the required property (e.g. strong normalization) and then to develop semantics in order to obtain the soundness of the type assignment. A consequence of soundness, the fact that every term typable by a type in the type system belongs to the interpretations of that type, leads to the fact that terms typable in the type system satisfy the required property, since the type interpretations are built up in that way.

In order to develop the reducibility method we consider the applicative structure whose domain are the terms in Λx and where the application is just the application of terms.

Definition 6.1 (REDUCIBLE TERMS)

i) We define the collection of set of terms $\mathcal{R}^\rho$ inductively over types by:

$$\begin{array}{rcl} \mathcal{R}^\varphi & = & \mathcal{SN} \\ \mathcal{R}^{\sigma\to\tau} & = & \{M \mid \forall N \in \mathcal{R}^\sigma\,[MN \in \mathcal{R}^\tau]\} \\ \mathcal{R}^{\sigma\cap\tau} & = & \mathcal{R}^\sigma \cap \mathcal{R}^\tau. \end{array}$$

ii) We define the set $\mathcal{R}$ of *reducible terms* by: $\mathcal{R} = \{M \mid \exists \rho\,[M \in \mathcal{R}^\rho]\} = \bigcup_{\rho\,\in\,\mathcal{T}} \mathcal{R}^\rho$.

Notice that, if $M \in \mathcal{R}^\sigma$, not necessarily there exists a Γ such that $\Gamma \vdash M{:}\sigma$. For example, if φ, φ' are two different type variables, then $\lambda x.x \in \mathcal{R}^{\varphi\to\varphi'}$, since $(\lambda x.x)M \in \mathcal{SN}$ whenever $M \in \mathcal{SN}$, but we cannot derive $\emptyset \vdash \lambda x.x{:}\varphi{\to}\varphi'$. Also, since $\lambda x.x \in \mathcal{SN}$, $\lambda x.x \in \mathcal{R}^\varphi$, but we cannot derive $\emptyset \vdash \lambda x.x{:}\varphi$.

We now show that reducibility implies strong normalization and that all term-variables are reducible. For the latter, it is convenient to show a generalization: all typable strongly normalisable terms that start with a term variable are reducible.

Lemma 6.2 *i)* $\mathcal{R} \subseteq \mathcal{SN}$.
ii) $x\vec{N} \in \mathcal{SN} \;\Rightarrow\; \forall \rho\,[x\vec{N} \in \mathcal{R}^\rho]$.

Proof: By simultaneous induction on the structure of types.

i) (φ) : By Def. 6.1.
$(\sigma{\to}\tau)$: $M \in \mathcal{R}^{\sigma\to\tau} \Rightarrow (IH(ii))\;\; M \in \mathcal{R}^{\sigma\to\tau}\;\&\; x \in \mathcal{R}^\sigma \Rightarrow (6.1)$
$\qquad\qquad Mx \in \mathcal{R}^\tau \Rightarrow (IH(i))\;\; Mx \in \mathcal{SN} \;\Rightarrow\; M \in \mathcal{SN}$.
$(\sigma\cap\tau)$: $M \in \mathcal{R}^{\sigma\cap\tau} \Rightarrow (6.1)\;\; M \in \mathcal{R}^\sigma\;\&\; M \in \mathcal{R}^\tau \Rightarrow (IH(i))\;\; M \in \mathcal{SN}$.

ii) (φ) : $x\vec{N} \in \mathcal{SN} \Rightarrow (6.1)\;\; x\vec{N} \in \mathcal{R}^\varphi$.
$(\sigma{\to}\tau)$: $\begin{array}{ll} x\vec{N} \in \mathcal{SN} & \Rightarrow\; (3.2, (\textsf{gen-var})) \\ \forall M \in \mathcal{SN}\,[x\vec{N}M \in \mathcal{SN}] & \Rightarrow\; (IH(i)) \\ \forall M \in \mathcal{R}^\sigma\,[x\vec{N}M \in \mathcal{SN}] & \Rightarrow\; (IH(ii)) \\ \forall M \in \mathcal{R}^\sigma\,[x\vec{N}M \in \mathcal{R}^\tau] & \Rightarrow\; (6.1)\; x\vec{N} \in \mathcal{R}^{\sigma\to\tau} \end{array}$
$(\sigma\cap\tau)$: $x\vec{N} \in \mathcal{SN} \Rightarrow (IH(ii))\;\; x\vec{N} \in \mathcal{R}^\sigma\;\&\; x\vec{N} \in \mathcal{R}^\tau \Rightarrow (6.1)\;\; x\vec{N} \in \mathcal{R}^{\sigma\cap\tau}$. ■

We now show that all sets $\mathcal{R}^\rho$ are closed under the rules (subs), (gen-B), (gen-App), (gen-Abs), (gen-I) and (gen-gc). This result is needed in the proof of Theorem 6.5.

Lemma 6.3 (SATURATION) *For all ρ the sets $\mathcal{R}^\rho$ are $\mathcal{SN}$-saturated.*

Proof: All these closures are shown by induction on the structure of types. For the case of a type-variable, $\mathcal{R}^\varphi = \mathcal{SN}$, which is $\mathcal{SN}$-saturated (Theorem 3.4). For the rest of the induction, since the proofs are all very similar, we will not show all in detail, but focus on rule (subs). Then:

$(\sigma{\to}\tau)$: $\begin{array}{ll} (P\overrightarrow{\langle x{=}N\rangle}\,\langle y{=}Q\overrightarrow{\langle x{=}N\rangle}\rangle)\overrightarrow{M} \in \mathcal{R}^{\sigma\to\tau} & \Rightarrow\; (6.1) \\ \forall R \in \mathcal{R}^\sigma\,[(P\overrightarrow{\langle x{=}N\rangle}\,\langle y{=}Q\overrightarrow{\langle x{=}N\rangle}\rangle)\overrightarrow{M}R \in \mathcal{R}^\tau] & \Rightarrow\; (IH) \\ \forall R \in \mathcal{R}^\sigma\,[((P\langle y{=}Q\rangle)\overrightarrow{\langle x{=}N\rangle})\overrightarrow{M}R \in \mathcal{R}^\tau] & \Rightarrow\; (6.1) \\ ((P\langle y{=}Q\rangle)\overrightarrow{\langle x{=}N\rangle})\overrightarrow{M} \in \mathcal{R}^{\sigma\to\tau}. & \end{array}$
$(\sigma\cap\tau)$: Immediate by Def. 6.1 and induction. ■

We shall prove our strong normalization result by showing that every typable term is reducible. For this, we need to prove a stronger property: we will show that if we substitute term-variables by reducible terms in a typable term, then we obtain a reducible term. This gives the soundness of our type interpretation.

Theorem 6.4 (SOUNDNESS) *If* $\{(x_1{:}\mu_1), \ldots, (x_n{:}\mu_n)\} \vdash M{:}\sigma$*,and, for* $i \in \underline{n}$, $N_i \in \mathcal{R}^{\mu_i}$, *then* $M\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^\sigma$.

Proof: Note that by the convention on variables we may assume that for all $1 \leq i, j \leq n$, $x_i \notin \mathit{fv}(N_j)$. The proof is by induction on the structure of derivations. We will use the $\mathcal{SN}$-saturation of the saturated sets (Lemma 6.3) just mentioning the rule names. Let $\Gamma = \{(x_1{:}\mu_1), \ldots, (x_n{:}\mu_n)\}$.

(start) : Then $M \equiv x_j$, and $\mu_j = \sigma$, for some $j \in \underline{n}$. Since $N_j \in \mathcal{R}^{\mu_j}$, $N_j \in \mathcal{R}^{\sigma}$. Then, by rules (gen-I) and (gen-gc), $x_j\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

($\rightarrow$I) : Then $M \equiv \lambda y.M'$, $\sigma = \rho{\rightarrow}\tau$, and $\Gamma, (y{:}\rho) \vdash M'{:}\tau$. Let $N \in \mathcal{R}^{\rho}$, then, by induction, $M'\,\overrightarrow{\langle x{=}N\rangle}\,\langle y{=}N\rangle \in \mathcal{R}^{\tau}$. So, by rule (gen-B), $(\lambda y.M'\,\overrightarrow{\langle x{=}N\rangle})N \in \mathcal{R}^{\tau}$, and, by Def. 6.1, $\lambda y.M'\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\rho\rightarrow\tau}$. We can assume $y \notin \mathit{fv}(N)$, so, by rule (gen-Abs), $(\lambda y.M')\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\rho\rightarrow\tau}$.

($\rightarrow$E) : Then $M \equiv M_1 M_2$ and there exists τ such that $\Gamma \vdash M_1{:}\tau{\rightarrow}\sigma$ and $\Gamma \vdash M_2{:}\tau$. By induction, $M_1\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\tau\rightarrow\sigma}$ and $M_2\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\tau}$. But then, by Definition 6.1, $M_1\,\overrightarrow{\langle x{=}N\rangle}\,M_2\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$, so, by rule (gen-App), $(M_1 M_2)\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

($\cap$I) : Then $\sigma \equiv \sigma_1 \cap \sigma_2$ and, for $i \in \underline{2}$, $\Gamma \vdash M{:}\sigma_i$. So, by induction, $M\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma_1}$ and $M\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma_2}$, so, by Def. 6.1, $M\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

($\cap$E) : Then there exists τ such that $\Gamma \vdash M{:}\sigma\cap\tau$, and, by induction, $M\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma\cap\tau}$. Then, by Def. 6.1, $M\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

(cut) : Here $M \equiv P\langle y{=}Q\rangle$, and there exists τ such that $\Gamma, (y{:}\tau) \vdash P{:}\sigma$ and $\Gamma \vdash Q{:}\tau$. Then, by induction applied to the right-hand hypothesis, $Q\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\tau}$. Then again by induction, on the left-hand hypothesis, $P\,\overrightarrow{\langle x{=}N\rangle}\,\langle y{=}Q\,\overrightarrow{\langle x{=}N\rangle}\rangle \in \mathcal{R}^{\sigma}$. So, by rule (subs), $(P\langle y{=}Q\rangle)\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

(drop) : Here $M \equiv P\langle y{=}Q\rangle$, $\Gamma \vdash P{:}\sigma$, $y \notin \Gamma$ and there exist Δ, τ such that $\Delta \vdash Q{:}\tau$. By induction $P\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$. We may be sure that $Q \in \mathcal{SN}$ by the induction hypothesis applied to the derivation from Δ. Since $y \notin \mathit{av}(P)$ we may use closure of $\mathcal{R}^{\sigma}$ under rule (gen-gc) to conclude that $(P\langle y{=}Q\rangle)\,\overrightarrow{\langle x{=}N\rangle} \in \mathcal{R}^{\sigma}$.

(K-cut) : The proof is very similar to the (drop) case, or we may appeal to Theorem 4.8. ■

Theorem 6.5 *If $\Gamma \vdash M{:}\sigma$ for some Γ, σ then $M \in \mathcal{SN}$.*

Proof: Suppose Γ is $\{(x_1{:}\rho_1), \ldots, (x_m{:}\rho_m)\}$. By Lemma 6.2(*ii*), all term-variables are reducible for any type, so, by Theorem 6.4, for all M, $M\,\overrightarrow{\langle x{=}y\rangle}$ is reducible, where $\overrightarrow{y}$ are fresh. By Lemma 6.2 (*i*) the term $M\,\overrightarrow{\langle x{=}y\rangle}$ is strongly normalizing, and since M is a subterm of this the result follows. ■

7 Characterizing weak normalization and head normalization

The system $\mathcal{E}$ is obtained from the system $\mathcal{D}$ of [17] by adding the rules (drop) and (K-cut). The system $\mathcal{D}_\omega$ is the extension of $\mathcal{D}$ obtained by adding a universal type ω; in [17] characterizations of the head-normalizing and left-most-normalizing terms of $\lambda\mathtt{x}$ were obtained in terms of typability in $\mathcal{D}_\omega$.

The main result of this paper is that typability in system $\mathcal{E}$ serves to characterize the strongly-normalizing terms of $\lambda\mathtt{x}$, and therefore that the rules (drop) and (K-cut) capture this important aspect of reduction in explicit substitutions calculi. But a natural question to raise at this point is whether the addition of rules (drop) and (K-cut) behaves well in the presence of a universal type. In particular, we may ask whether the normalization theorems of [17] still hold in the presence of the new rules. In this section we show that they do continue to hold. That is, we will verify that the $\mathcal{D}_\omega$-characterizations of normalizing and head-normalizing terms from [17] generalize in the natural way to $\mathcal{E}_\omega$. The first observation is that when a universal type is added to $\mathcal{E}$ the resulting system is equivalent to $\mathcal{D}_\omega$.

7.1 Extending the type system

Definition 7.1 The type system $\mathcal{E}_\omega$ is obtained from system $\mathcal{E}$ by adding the type constant ω and the rule:

$$(\omega\mathsf{I}) \quad \frac{}{\Gamma \vdash M : \omega}$$

The type system $\mathcal{D}_\omega$ is obtained by adding ω and rule $(\omega\mathsf{I})$ to the system $\mathcal{D}$ of [17].

Theorem 7.2 *Suppose $\Gamma \vdash M : \tau$ in system $\mathcal{E}_\omega$. Then $\Gamma \vdash M : \tau$ in system $\mathcal{D}_\omega$ as well.*

Proof: By induction over typing derivations. In light of the equivalence between (drop) and $(\mathsf{K\text{-}cut})$ it suffices to show that an application of rule (drop) can be simulated in $\mathcal{D}_\omega$. So suppose

$$(\mathsf{drop}) \quad \frac{\Gamma \vdash M : \tau \quad \Delta \vdash N : \sigma}{\Gamma \vdash M\langle x = N\rangle : \tau}\,(x \notin \mathit{av}(M))$$

By induction we can derive $\Gamma \vdash M : \tau$ in $\mathcal{D}_\omega$, so certainly $\Gamma, (x{:}\omega) \vdash M : \tau$. By $(\omega\mathsf{I})$, $\Gamma \vdash N : \omega$ in $\mathcal{D}_\omega$, so we have

$$(\mathsf{cut}) \quad \frac{\Gamma, (x{:}\omega) \vdash M : \tau \quad \Gamma \vdash N : \omega}{\Gamma \vdash M\langle x = N\rangle : \tau}$$

in $\mathcal{D}_\omega$, as desired.

7.2 Head reduction and left-most reduction

The head and left-most redexes from the classical λ-calculus appear in $\lambda\mathtt{x}_{gc}$ as head or left-most B-redexes. But the general notions of head or left-most redex in $\lambda\mathtt{x}_{gc}$ must take into account the rules for applying substitutions. In fact, the correct definitions of head and left-most reduction are more subtle than in the classical calculus. Essentially this is because $\lambda\mathtt{x}_{gc}$ has a critical pair, due to the following overlapping reductions:

$$(\lambda x.M)\langle y = L\rangle\ N\langle y = L\rangle \longleftarrow ((\lambda x.M)N)\langle y = L\rangle \longrightarrow M\langle x = N\rangle\langle y = L\rangle$$

Each of the reductions above has a claim on our intuition for being considered a "head reduction." In fact we consider them each to be head reductions.

Definition 2 (HEAD REDUCTION) *Head reduction* is the closure of the rules of $\lambda\mathtt{x}_{gc}$ (Definition 2.4) under the structural rules of Table 3.

A term M is *head normalizing* if there is no infinite head-reduction starting from M. The set of head normalizing terms is denoted $\mathcal{HN}$.

Definition 3 (LEFT-MOST REDUCTION) *Left-most reduction* is the closure of the rules of $\lambda\mathtt{x}_{gc}$ under the structural rules in Table 4.

A term M is *left-most normalizing* if there is no infinite left-most reduction starting from M. The set of left-most-normalizing terms is denoted $\mathcal{LN}$.

Observe that, in contrast to the classical notions, both head reduction and left-most reduction are non-deterministic strategies. Indeed both reductions out of the critical pair noted earlier count as head reductions.

For example, let T be $((\lambda x.M)N)\langle y = L\rangle$. Then T can rewrite by left-most reduction either to $P \equiv M\langle x = N\rangle\langle y = L\rangle$, or (in two steps) to $Q \equiv (\lambda x.M\langle y = L\rangle)\ N\langle y = L\rangle)$. Then, since $\lambda x.M\langle y = L\rangle$ is an abstraction, Q left-most-rewrites via rule B leading to $Q' \equiv M\langle y = L\rangle\langle x = N\langle y = L\rangle\rangle$.

$$\frac{M \xrightarrow{h} M' \quad M \text{ not an abstraction}}{MN \xrightarrow{h} M'N}$$

$$\frac{M \xrightarrow{h} M'}{\lambda x.M \xrightarrow{h} \lambda x.M'}$$

$$\frac{M \xrightarrow{h} M' \quad M \text{ not an abstraction}}{M\langle x=A\rangle \xrightarrow{h} M'\langle x=A\rangle}$$

Table 3: Head reduction

$$\frac{M \xrightarrow{l} M' \quad M \text{ not an abstraction}}{MN \xrightarrow{l} M'N} \qquad \frac{M \xrightarrow{l} M'}{\lambda x.M \xrightarrow{l} \lambda x.M'}$$

$$\frac{M \xrightarrow{l} M' \quad M \text{ not an abstraction}}{M\langle x=N\rangle \xrightarrow{l} M'\langle x=N\rangle} \qquad \frac{M_i \xrightarrow{l} M_i' \quad M_i \text{ left-most non-normal}}{xM_1 \ldots M_i \ldots M_n \xrightarrow{l} xM_1 \ldots M_i' \ldots M_n}$$

Table 4: Left-most reduction

7.3 Characterization theorems

We will assume familiarity with [17] in this subsection; we derive the characterization theorems by indicating how to lift the results of that paper. There is a technical issue to be dealt with, however: the garbage-collection rule (gc) in the current paper is more liberal than the traditional rule in the system of [17]. In this section we refer to the traditional garbage-collection rule as gc^-:

$$M\langle x=N\rangle \quad \longrightarrow \quad M, \text{ if } x \notin fv(M) \quad (\mathsf{gc}^-)$$

Formally, since [17], treats a different reduction system, it is difficult to quote results there in support of results about the system of this paper. But the *arguments* of the first paper carry over almost word-for-word. In light of this we have chosen to indicate below precisely where the distinction between the systems makes a difference, rather than repeating the entire development.

The following definitions are due to Cardone and Coppo [9]: A type is *proper* if it has no positive occurrence of ω. A type is *trivial* if it can be generated by the following rules:

i) ω is trivial,

ii) If σ is trivial and τ is any type, then $\tau \to \sigma$ is trivial,

iii) If σ and τ are trivial, then $\sigma \cap \tau$ is trivial.

The following lemma isolates the place where we must ackowledge the difference in garbage-collection rules.

Lemma 4 *If M is typable with a non-trivial type in system $\mathcal{D}_\omega$ then M is head-normalizing in the calculus $\lambda\mathtt{x}_{gc}$.*

If M is typable in system $\mathcal{D}_\omega$ with a type not involving ω then M is left-most-normalizing in the calculus $\lambda\mathtt{x}_{gc}$.

Proof: Each of these assertions is proved in [17] for the system $\lambda\mathtt{x}_{\mathsf{gc}^-}$ (Theorems 8.1 and 8.2 there). We invite the reader to check that in that paper, the only places where the garbage-collection rule is analyzed are Lemmas 3.2 and 3.5 and that the proofs of each of these Lemmas are essentially unchanged if the current, more liberal, gc rule is used. The rest of the development in [17] is unchanged, completing the proof.

Theorem 7.3 *Let M be a closed term. The following are equivalent.*

i) M is typable with a non-trivial type in system $\mathcal{E}_\omega$.
ii) M is head-normalizing in the calculus $\lambda\mathtt{x}_{gc}$.
iii) M is head-normalizing in the calculus $\lambda\mathtt{x}$ (without garbage-collection).
iv) M has a head normal form.
v) M is solvable, that is, there is an n and terms $X_1, \ldots X_n$ such that $MX_1 \cdots X_n = \lambda x.x$.

Proof: By Theorem 7.2 we may replace, in (*i*), "$\mathcal{E}_\omega$" by "$\mathcal{D}_\omega$." Then each of the equivalences has been proved in [17] with the exception of the implication from (*i*) to (*ii*) since, in [17] garbage-collection refers to the more restricted rule gc^-. But for this impication we here use Lemma 4.

Theorem 7.4 *Let M be a closed term. The following are equivalent.*

i) M is typable in system $\mathcal{E}_\omega$ with a type not involving ω.
ii) M is typable with a proper type in system $\mathcal{E}_\omega$.
iii) M is left-most-normalizing in the calculus $\lambda\mathtt{x}_{gc}$.
iv) M is left-most-normalizing in the calculus $\lambda\mathtt{x}$ (without garbage-collection).
v) M has a normal form.

Proof: As for Theorem 7.3.

In Theorem 7.4, the implications (*v*) to (*iii*) and (*v*) to (*iv*) state that in $\lambda\mathtt{x}$ and $\lambda\mathtt{x}_{gc}$ left-most reduction is a normalizing strategy.

8 Conclusion

We have defined an improved system of intersection types for calculi of explicit substitutions and shown that it characterizes the strongly normalizing terms. The new rules allowing us to type all strongly normalizing terms are consistent with the addition of a universal type, in the sense that the characterizations of head- and left-most-normalizing terms obtained in previous work are still valid in the extended system.

The new notion of *available* variable occurrence plays an important role in the type system, and indeed allows us to define a more powerful notion of garbage collection than has appeared elsewhere in the explicit substitutions literature. We like to remark the similarity between the reduction rule (gc) and the classical mark-and-sweep algorithm for garbage collection. As a matter of fact the computation of the set of available variables of a term corresponds to the mark phase, while the reduction using only rule (gc) corresponds to the sweep phase. Notice that this is not true for the similar rules of [8, 17]. We think that it could be interesting to investigate the use of the garbage collection based on availability of variables in the implementations of functional programming languages.

Acknowledgements

The authors are grateful to Norman Danner, Fr´ed´eric Lang, Simona Ronchi della Rocca, and Kristoffer Rose for many helpful discussions and the referees of the conferences LATIN'02 and TCS'02 for helpful comments.

References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. L´evy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[2] R. Amadio and P.-L. Curien. *Domains and lambda-calculi*. Cambridge University Press, 1998.

[3] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[4] H. P. Barendregt. Lambda calculi with types. In S. Abramsky, D. M. Gabby, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 2, pages 117–309. Oxford University Press, 1992.

[5] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, 1996.

[6] R. Bloo. *Preservation of Termination for Explicit Substitution*. PhD thesis, Technische Universiteit Eindhoven, 1997. IPA Dissertation Series 1997-05.

[7] R. Bloo and J. H. Geuvers. Explicit substitution: on the edge of strong normalization. *Theoretical Computer Science*, 211:375 – 395, 1999.

[8] R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN '95*, pages 62–72, 1995.

[9] F. Cardone and M. Coppo. Two extension of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science*, volume 31 of *APIC Series*, pages 19–75. Academic Press, New York, NY, 1990.

[10] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.

[11] 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.

[12] N. G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. TH-Report 78-WSK-03, Technological University Eindhoven, Netherlands, Department of Mathematics, 1978.

[13] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[14] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterization of lambda -terms using intersection types. In M.Nielsen and B.Rovan, editors, *MFCS'00*, volume 1893 of *Lecture Notes in Computer Science*, pages 304–314. Springer-Verlag, 2000.

[15] R. Di Cosmo and D. Kesner. Strong normalization of explicit substitutions via cut elimination in proof nets. In *LICS'97*, pages 35–46. IEEEC Society Press, 1997.

[16] D. Dougherty and P. Lescanne. Reductions, intersection types, and explicit substitutions (extended abstract). In S. Abramsky, editor, *TLCA'01*, volume 2044 of *Lecture Notes in Computer Science*, pages 121–135. Springer-Verlag, 2001.

[17] D. Dougherty and P. Lescanne. Reductions, intersection types, and explicit substitutions. *Mathematical Structures in Computer Science*, to appear.

[18] A. G. Dragalin. *Mathematical Intuitionism: Introduction to Proof Theory*, volume 67 of *Translations of Mathematical Monographs*. American Mathematical Society, 1987.

[19] J. Gallier. Typing untyped lambda terms, or reducibility strikes again. *Ann. Pure Appl. Logic*, 91:231–270, 1998.

[20] S. Ghilezan. Strong normalization and typability with intersection types. *Notre-Dame Journal of Formal Logicrr*, 37(1):44–52, 1996.

[21] J. Goubault-Larrecq. Lambda-calcul, logique et machines. École Normale Sup´erieure de Cachan, 2001.

[22] H. Herbelin. Explicit substitutions and reducibility. *Journal of Logic and Computation*, 11(3):429–449, 2001.

[23] F. Kamareddine and A. R´ıos. Extending a lambda-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, 1997.

[24] J.-L. Krivine. *Lambda-calcul Types et modèles*. Masson, Paris, 1990.

[25] J.-L. Krivine. *Lambda calculus, types and models*. Ellis Horwood, 1993.

[26] D. Leivant. Typing and computational properties of lambda expressions. *Theoretical Computer Science*, 44(1):51–68, 1986.

[27] S. Lengrand, D. Dougherty, and P. Lescanne. An improved system of intersection types for explicit substitutions. In R. Baeza-Yates, U. Montanari, and N. Santoro, editors, *Foundations of Information Technology in the era of Network and Mobile Computing, IFIP Congress*, pages 511–524. Kluwer Academic Publishers, 2002.

[28] P. Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In Hans-J. Bôhm, editor, *POPL'94*, pages 60–69. ACM Press, 1994.

[29] P. Lescanne and J. Rouyer-Degli. The calculus of explicit substitutions $\lambda\upsilon$. Technical Report RR-2222, INRIA-Lorraine, January 1994.

[30] P.-A. Melliès. Typed λ-calculi with explicit substitution may not terminate. In M. Dezani and G.Plotkin, editors, *TLCA'95*, volume 902 of *Lecture Notes in Computer Science*, pages 328–334. Springer-Verlag, 1995.

[31] J. C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 415–431. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1990.

[32] J. C. Mitchell. *Foundation for Programmimg Languages*. MIT Press, 1996.

[33] G. Pottinger. A type assignment for the strongly normalizable λ-terms. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–578. Academic Press, 1980.

[34] E. Ritter. Characterising explicit substitutions which preserve termination. In J.-Y.Girard, editor, *TLCA'99*, volume 1581 of *Lecture Notes in Computer Science*, pages 325–339. Springer-Verlag, 1999.

[35] K.H. Rose. *Operational Reduction Models for Functional Programming Languages*. PhD thesis, DIKU, Universitetsparken 1, DK-2100 København Ø, February 1996. DIKU report 96/1.

[36] W. W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.

[37] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

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

[39] S. van Bakel and M. Dezani-Ciancaglini. Characterizing strong normalization for explicit substitutions. In S.Rajsbaum, editor, *LATIN'02*, volume 2286 of *Lecture Notes in Computer Science*, pages 356–370. Springer-Verlag, 2002.