Intersection types for explicit substitutions

Stéphane Lengrand [a] Pierre Lescanne [a] Dan Dougherty [b]
Mariangiola Dezani-Ciancaglini [c,1], Steffen van Bakel [d]

[a] *École Normale Supérieure de Lyon 46, Allée d'Italie, 69364 Lyon 07, FRANCE, E-mail:* {`Stephane.Lengrand,Pierre.Lescanne`}`@ens-lyon.fr`

[b]*Department of Computer Science, Worcester Polytechnic Institute Worcester, MA 101609 USA, E-mail:* `dd@cs.wpi.edu`

[c]*Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy, E-mail:* `dezani@di.unito.it`

[d]*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`,

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.

Key words: Calculi of explicit substitutions, intersection types, strong normalization.

1 Introduction

An explicit substitutions calculus is a refinement of the classical Lambda Calculus (LC) [6] 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

[1] Partially supported by EU within the FET - Global Computing initiative, project DART ST-2001-33477 , and MURST Projects COMETA and McTati. The funding bodies are not responsible for any use that might be made of the results presented here.

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 Lévy [1] and de Bruijn [12] defined the first calculi of explicit substitutions.

Intersection type disciplines originated in [14,15] to overcome the limitations of Curry's type assignment system and to provide a characterization of the strongly normalizing terms of the λ-calculus [36]. Since then, intersection types disciplines have been used in a series of papers for characterizing evaluation properties of λ-terms [29,28,3,4,23,2,22,17].

As discussed in [20], one can see an explicit substitution calculus as an improvement on both the system of combinators and LC, 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 LC. Observe that LC can be viewed as 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 LC, and the study of explicit substitutions represents a deeper examination of the relationship between abstraction and application.

A fundamental property of typed LC is strong normalization: no typable term admits an infinite reduction sequence. Melliès [33] made the somewhat surprising discovery that strong normalization fails even for simply-typed terms of the Abadi-Cardelli-Curien-Lévy calculus.

Given the central place that strong normalization occupies in the theory and application of LC, it is important to study this property in systems of explicit substitutions. Melliés' 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 [37]. 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 [31,11,26,10,8], among other works. Here we work in the composition-free calculus $\lambda\mathtt{x}$ [11] and a calculus $\lambda\mathtt{x}_{\mathsf{gc}}$ 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 [11].

Previous work [19,20] 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 by means of typability. But it was shown in [19] 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 *Consider the terms*

$$M_1 \equiv ((\lambda y.z)(xx))\langle x{=}\lambda a.aa\rangle \text{ and}$$
$$M_2 \equiv z\langle y{=}xx\rangle\langle x{=}\lambda a.aa\rangle$$

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 [19]: 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 [19].

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 [19] 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 [5,30]. 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 [5], 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 the notion of *available* variable occurrence in a term (Definition 3). This is a refinement of the notion of free variable, first considered in [11] (Remark 2.3).

The present paper is an joint expanded version of the conference papers [5] and [30]; we present both rules in a uniform system, and investigate the relationship between the two systems.

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 λx 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 [32,8,11,9,10,38,18]. It follows that the classical intersection type 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 [25] has proposed also a direct proof of strong normalization for a simply typed calculus of explicit substitution which interprets a sequent calculus (he restricts the 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 an account of the *cut rule* [21].

Recently we learned that Jean Goubault-Larrecq proposes, in the exercises of his course [24], a type system with intersection types for (a version with De Bruijn indices of) the calculus of explicit substitutions $\lambda\upsilon$ introduced in [31]. Each typable term in this calculus is shown to be strongly normalizing, but the converse 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-admissibility 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 [19] extend to system $\mathcal{E}_\omega$.

Notation. Our notation is consistent with that of [7], to which we refer the reader for background on LC. We will use $\underline{n}$ for $\{1, \ldots, n\}$.

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

2.1 Syntax and available variables

Definition 2 *The set $\lambda\mathtt{x}$ of terms with explicit substitutions is defined as follows :*

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

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:

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

We will see in Figure 2 another description of the set of terms with explicit substitutions called the *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 3 *The* free *variables in a term are:*

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

A variable occurrence which is not free is called a bound *occurrence. The* available *variables in a term are:*

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

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, and that free and available variables coincide for pure terms.

Lemma 4 $\mathit{av}(M) \subseteq \mathit{fv}(M)$.

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, that, in the term $z\langle y{=}xx\rangle$, the variable x is free, but not available.

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 α-conversion. Throughout this paper, we will assume the Barendregt convention on variables [6] 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 20) 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 5 ($\lambda\mathtt{x}$ and $\lambda\mathtt{x}_{\mathsf{gc}}$) *We identify the following reduction rules on $\lambda\mathtt{x}$ terms.*

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

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 $\lambda\mathtt{x}_{\mathsf{gc}}$ 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 [11,20], in that it uses availability of the variable instead of freeness. This models a more liberal rule for garbage collection.

When it is clear from the context which notion of reduction is used, $\longrightarrow$ will denote the reduction relation and $\twoheadrightarrow$ will denote its reflexive and transitive closure.

The following lemma justifies the addition of our rule (gc) to $\lambda\mathtt{x}$.

Lemma 6 *If $x \notin \mathit{av}(M)$ then*

$$M\langle x{=}N\rangle =_{\mathtt{x}} M$$

*where $\cdot =_{\mathtt{x}} \cdot$ is the equivalence relation on terms generated by rules (*App*), (*Abs*), (*VarI*), (*VarK*).*

PROOF. By induction on the structure of terms.

Cases $M \equiv y, \lambda y.P, PQ$ are straightforward.

For the remaining case, $M \equiv P\langle y{=}Q\rangle$, first of all, notice that, by Corollary 2.13 and Proposition 2.14(a) of [11], we have

$$(P\langle y{=}Q\rangle)\langle x{=}N\rangle =_{\mathtt{x}} (P\langle x{=}N\rangle)\langle y{=}Q\langle x{=}N\rangle\rangle.$$

We distinguish two cases:

- $y \in \mathit{av}(P)$. Then $x \notin (\mathit{av}(P)\backslash\{y\}) \cup \mathit{av}(Q)$, so

$$\begin{array}{ll} (P\langle x{=}N\rangle)\langle y{=}Q\langle x{=}N\rangle\rangle & =_{\mathtt{x}} (\mathit{IH}) \\ P\langle y{=}Q\langle x{=}N\rangle\rangle & =_{\mathtt{x}} (\mathit{IH})\ P\langle y{=}Q\rangle. \end{array}$$

- $y \notin \mathit{av}(P)$; notice that then also $P\langle y{=}Q\rangle =_{\mathtt{x}} P$, by induction. Then $x \notin \mathit{av}(P)$, so

$$\begin{array}{ll} (P\langle x{=}N\rangle)\langle y{=}Q\langle x{=}N\rangle\rangle & =_{\mathtt{x}} (\mathit{IH}) \\ P\langle y{=}Q\langle x{=}N\rangle\rangle & =_{\mathtt{x}} (\mathit{IH})\ P \end{array}$$

  So $(P\langle x{=}N\rangle)\langle y{=}Q\langle x{=}N\rangle\rangle =_{\mathtt{x}} P\langle y{=}Q\rangle$.

In particular, for both cases, we get $(P\langle y{=}Q\rangle)\langle x{=}N\rangle =_{\mathtt{x}} P\langle y{=}Q\rangle$. ■

By induction on reductions one can check that the set of available variables does not increase when terms are reduced.

Lemma 7 *(1) If $M \longrightarrow N$ then $\mathit{av}(M) \supseteq \mathit{av}(N)$.*
(2) If $x \notin \mathit{av}(M)$, $M \longrightarrow N$ and N is a pure term then $x \notin \mathit{fv}(N)$.

In contrast with LC 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). Specifically, a term of the form

$$((\lambda x.M)N)\langle y{=}L\rangle$$

can be reduced to either of

$$(\lambda x.M)\langle y{=}L\rangle\ N\langle y{=}L\rangle \quad \text{or} \quad M\langle x{=}N\rangle\langle y{=}L\rangle$$

Most of the difficulty in working with the system is due to this critical pair, as we will see.

Definition 8 ($\mathcal{SN}$) *We say, as usual, that M is* in normal form *if M is redex free, and write nf(M) if M is in normal form. M is* normalisable *is there exists M' in normal form such that $M \longrightarrow\!\!\!\!\!\rightarrow M'$, 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 λx.*

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 12), 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 [20].

Lemma 9 *The set $\mathcal{SN}$ is closed under rule:*

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

Figure 1 tells us how the set of strongly normalizing terms can be generated by induction. Rule $(\mathsf{gen\text{-}var})$ has a number (possibly zero) of terms as upper part. The rule $(\mathsf{gen\text{-}App})$ is interesting, as a first example of the role of our critical pair. When the term (UV) is in fact a B-redex, it is not obvious that this rule is sound, that is, that pushing the substitution through the application (as opposed to firing the B-redex) preserves the existence of an infinite reduction. But in fact it is sound, as we will see.

Proposition 10 *$\mathcal{SN}$ is* generated *by the rules of Figure 1.*

PROOF. We first show that the rules in Figure 1 generate *only* terms in $\mathcal{SN}$, i.e.

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

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

$$(\mathsf{gen\text{-}App}) : \frac{(U\langle x{=}N\rangle)(V\langle x{=}N\rangle)\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}}{(UV)\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}}$$

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

Fig. 1. Generation of $\mathcal{SN}$

that for each rule, if the upper term(s) belong to $\mathcal{SN}$ then the lower term belong to $\mathcal{SN}$.

We only consider two of the rules: (gen-I), because it is typical, and (gen-App), because it uses techniques specific to this set of rules.

(gen-I) : Suppose $N\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}$ is in $\mathcal{SN}$. Suppose, towards a contradiction, that $x\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}$ 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 $N\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}$ is not in $\mathcal{SN}$, which is a contradiction.
- or this reduction is of the form

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

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

(gen-App) : Suppose $(UV)\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}$ is not in $\mathcal{SN}$, then there exists an infinite reduction starting from this term. If this reduction never reduces 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, and $(U\langle x{=}N\rangle)(V\langle x{=}N\rangle)\overrightarrow{\langle z{=}Q\rangle}$ is not in

$\mathcal{SN}$, which is a contradiction. If

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

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

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

But the assumption is that $(U\langle x{=}N\rangle)(V\langle x{=}N\rangle)\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}$ is in $\mathcal{SN}$, so also $U'\langle x{=}N'\rangle\langle y{=}V'\langle x{=}N'\rangle\rangle\overrightarrow{\langle z{=}Q'\rangle}\overrightarrow{P'}$ is in $\mathcal{SN}$. Therefore, by (Lemma 9), applying $(\textsf{subs})$ gives that $(U'\langle y{=}V'\rangle\langle x{=}N'\rangle)\overrightarrow{\langle z{=}Q'\rangle}\overrightarrow{P'}$ in $\mathcal{SN}$, which is a contradiction.

To conclude the proof, we need to show that the rules in Figure 1 generate *all* the terms in $\mathcal{SN}$. This is proven by a double induction on the length of the longest derivation to normal form and on the structure of terms. Notice that the terms in the conclusions of the given rules cover all possible shapes of terms in $\lambda\mathrm{x}$. Moreover, it is easy to see that for each rule in Figure 1, if the lower term belong to $\mathcal{SN}$ then the upper term(s) belong to $\mathcal{SN}$. The induction hypothesis applies since the the upper term(s) either can be obtained by reducing the lower term or they are subterms of the lower term. ■

3.2 Saturated sets

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

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

Definition 11 *A set closed under the rules* $(\textsf{subs})$, $(\textsf{gen-B})$, $(\textsf{gen-Abs})$, $(\textsf{gen-App})$, $(\textsf{gen-I})$ *and* $(\textsf{gen-gc})$ *is said to be* $\mathcal{SN}$-saturated.

Theorem 12 (Saturation of $\mathcal{SN}$) *The set $\mathcal{SN}$ is $\mathcal{SN}$-saturated.*

PROOF. Because of Lemma 9 and Proposition 10 we need only to show that $\mathcal{SN}$ is closed under the new rule. To show closure under rule $(\textsf{gen-gc})$, we reformulate the proof of [20] to take into account the change from $\mathit{fv}(\cdot)$ to $\mathit{av}(\cdot)$ in the definition of $\lambda\mathsf{x}_{\mathsf{gc}}$. 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 $C[\![\overrightarrow{M_i}]\!]$ for short. We prove the following more general statement:

> *Let $C[\![\ldots]\!]$ be a multi-context, and N_i, M_i, $i\in\underline{n}$ 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 $\langle P,\boldsymbol{M},\boldsymbol{N}\rangle$, where P is a term, $\boldsymbol{M}$ and $\boldsymbol{N}$ are multisets of terms. Let $\sqsupset^m$ be the multiset extension [16] of $\sqsupset$, the converse of the proper subterm order, and let $\longrightarrow^m$ be the multiset extension of the reduction relation $\lambda\mathsf{x}_{\mathsf{gc}}$. The proof is by induction over the following relation:

$\langle P,\boldsymbol{M},\boldsymbol{N}\rangle\gg\langle P',\boldsymbol{M}',\boldsymbol{N}'\rangle$ if and only if

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

In what follows, P will be $C[\![\overrightarrow{M_i}]\!]$ and $\longrightarrow$ is well-founded out of P by hypothesis; $\boldsymbol{M}$ will be $\{M_1,\ldots,M_n\}$; $\boldsymbol{N}$ will be $\{N_1,\ldots,N_n\}$ and its $\lambda\mathsf{x}_{\mathsf{gc}}$-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 (4) and (5) below: there the term P 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 that $C[\![\overrightarrow{M_i}]\!]\in\mathcal{SN}$, and that $N_i\in\mathcal{SN}$ for $i\in\underline{n}$. We will prove that the term $C[\![\overrightarrow{M_i\langle x{=}N_i\rangle}]\!]$ reduces only to terms that are in $\mathcal{SN}$.

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

(2) $M_i\longrightarrow M_i'$: By induction.

(3) $N_j\longrightarrow N_j'$: Also by induction. Note that this case occurs only when the N_i are in $\mathcal{SN}$.

(4) $M_i=M_i^1M_i^2$ 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

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

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.

(5) $M_i = \lambda y.M'_i$ and $M_i\langle x{=}N_i\rangle \longrightarrow \lambda y.(M'_i\langle x{=}N_i\rangle)$:

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

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.

(6) $M_i\langle x{=}N_i\rangle \longrightarrow M_i$, which is always applicable being $x \notin av(M_i)$: Since

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

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 λx_{gc}.

3.3 A perpetual strategy

In what follows we will define a perpetual strategy for our calculus, which is an extension to λx of the strategy defined in [6], page 338. It is based on the reduction of perpetual redexes.

Definition 13 (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 :*

$$\begin{array}{ll} MN, & \textit{if } MN \textit{ itself is a redex} \\ \textit{the perpetual redex of } M, & \textit{if } M \textit{ is not a normal form} \\ \textit{the perpetual redex of } N, & \textit{otherwise} \end{array}$$

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

$$\begin{array}{ll} \textit{the perpetual redex of } N, & \textit{if } M \equiv y \neq x \textit{ and } N \textit{ is not a normal form} \\ \textit{the perpetual redex of } M, & \textit{if } M \textit{ is a closure} \\ M\langle x{=}N\rangle, & \textit{otherwise} \end{array}$$

Definition 14 (Perpetual strategy) *The* perpetual strategy *is the strategy that reduces always the perpetual redex. It is denoted by $\rightsquigarrow$.*

$$\begin{array}{rcll}
\lambda x.M & \rightsquigarrow & \lambda x.M', & \text{if } M \rightsquigarrow M' \quad (\mathsf{perp}\text{-}\lambda) \\
x\overrightarrow{P}M\overrightarrow{Q} & \rightsquigarrow & x\overrightarrow{P}M'\overrightarrow{Q}, & \text{if } \mathit{nf}(x\overrightarrow{P}) \text{ and } M \rightsquigarrow M' \quad (\mathsf{perp\text{-}var}) \\
(\lambda x.M)N\overrightarrow{P} & \rightsquigarrow & M\langle x{=}N\rangle\overrightarrow{P} & (\mathsf{perp\text{-}B}) \\
(UV)\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & \rightsquigarrow & (U\langle x{=}N\rangle)(V\langle x{=}N\rangle)\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & (\mathsf{perp\text{-}App}) \\
(\lambda y.M)\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & \rightsquigarrow & (\lambda y.M\langle x{=}N\rangle)\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & (\mathsf{perp\text{-}Abs}) \\
x\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & \rightsquigarrow & N\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & (\mathsf{perp\text{-}I}) \\
y\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & \rightsquigarrow & y\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}, & \text{if } \mathit{nf}(N) \quad (\mathsf{perp\text{-}K}) \\
y\langle x{=}N\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P} & \rightsquigarrow & y\langle x{=}N'\rangle\overrightarrow{\langle z{=}Q\rangle}\overrightarrow{P}, & \text{if } N \rightsquigarrow N' \quad (\mathsf{perp\text{-}clo})
\end{array}$$

Fig. 2. The perpetual strategy and the head-form taxonomy

Figure 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 ($\mathsf{perp\text{-}var}$) give two forms of irreducible terms when $\mathit{nf}(M)$ 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.*

Since each term contains at most one perpetual redex, the perpetual strategy is deterministic. Note that, in the case of $\lambda \mathsf{x}_{\mathsf{gc}}$, 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 normalizing, the perpetual strategy (to be really perpetual) has to reduce N instead of causing it to disappear.

Theorem 15 *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 [15] with a pre-order which takes the idempotence, commutativity and associativity of the intersection type constructor into account.

Definition 16 *The set of* types*, ranged over by $\sigma, \tau, \rho, \ldots$, is inductively defined as follows*

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

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

$$\begin{array}{rcl} \tau_1 \cap \tau_2 & \leq & \tau_1, \\ \tau_1 \cap \tau_2 & \leq & \tau_2, \end{array}$$

$$\textit{if } \sigma \leq \tau_1 \textit{ and } \sigma \leq \tau_2 \textit{ then } \sigma \leq \tau_1 \cap \tau_2$$

The pre-order defines the equivalence relation on types :

$$\tau \sim \sigma \ \textit{ if and only if } \tau \leq \sigma \textit{ and } \sigma \leq \tau$$

In the concrete syntax of types we give, as usual, $\cap$ 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 in the presence of the intersection type constructor.

Definition 17 *An* environment *is a partial assignment from variables to types, where each individual assignment is written $(x{:}\tau)$. Environments are partially ordered as follows.*

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

By abuse of notation, we write $x \in \Gamma$ for $(\exists \tau).(x{:}\tau) \in \Gamma$. The environment $\Gamma \backslash x$ 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 environments.

Definition 18
$$\begin{aligned} \Gamma_1 \sqcap \Gamma_2 \;=\; & \{(x{:}\tau) \mid (x{:}\tau) \in \Gamma_1 \;\&\; x \notin \Gamma_2\} \;\cup \\ & \{(x{:}\tau) \mid (x{:}\tau) \in \Gamma_2 \;\&\; x \notin \Gamma_1\} \;\cup \\ & \{(x{:}\tau_1{\cap}\tau_2) \mid (x{:}\tau_1) \in \Gamma_1 \;\&\; (x{:}\tau_2) \in \Gamma_2\} \\ \Gamma, (x{:}\tau) \;=\; & \Gamma\backslash x \cup \{(x{:}\tau)\} \end{aligned}$$

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 19 • $\Gamma_1 \sqcap \Gamma_2 \leq \Gamma_1$ *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. These will appear in the definition below as (drop) and $(\mathsf{K\text{-}cut})$.

Definition 20 (Type Assignment Rules) *The system $\mathcal{E}$ of* type assignment *for terms in $\lambda\mathtt{x}$ is defined as follows:*

$$(\mathsf{start})\; \frac{}{\Gamma \vdash x{:}\sigma}\;((x{:}\sigma) \in \Gamma) \qquad (\to\mathsf{I})\; \frac{\Gamma, (x{:}\sigma) \vdash M{:}\tau}{\Gamma \vdash \lambda x.M{:}\sigma{\to}\tau}$$

$$(\mathsf{cut})\; \frac{\Gamma, (x{:}\sigma) \vdash M{:}\tau \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash M\langle x\!=\!N\rangle{:}\tau} \qquad (\to\mathsf{E})\; \frac{\Gamma \vdash M{:}\sigma{\to}\tau \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash MN{:}\tau}$$

$$(\mathsf{drop})\; \frac{\Gamma \vdash M{:}\tau \quad \Delta \vdash N{:}\sigma}{\Gamma \vdash M\langle x\!=\!N\rangle{:}\tau}\;(x \notin \mathsf{av}(M)) \qquad (\cap\mathsf{I})\; \frac{\Gamma \vdash M{:}\sigma \quad \Gamma \vdash M{:}\tau}{\Gamma \vdash M{:}\sigma{\cap}\tau}$$

$$(\mathsf{K\text{-}cut})\; \frac{\Gamma \vdash M{:}\tau \quad \Delta \vdash N{:}\sigma}{\Gamma \vdash M\langle x\!=\!N\rangle{:}\tau}\;(x \notin \Gamma) \qquad (\cap\mathsf{E})\; \frac{\Gamma \vdash M{:}\sigma_1{\cap}\sigma_2}{\Gamma \vdash M{:}\sigma_i}\;(i \in \{1,2\})$$

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 [20] is obtained by removing the inference rules (drop) and $(\mathsf{K\text{-}cut})$: 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, this system

$$\dfrac{\dfrac{}{\{z{:}\mu\} \vdash z : \mu} \qquad \dfrac{\dfrac{\dfrac{}{\{x{:}(\rho{\to}\nu){\cap}\rho\} \vdash x : (\rho{\to}\nu){\cap}\rho}}{\{x{:}(\rho{\to}\nu){\cap}\rho\} \vdash x : \rho{\to}\nu} \quad \dfrac{\dfrac{}{\{x{:}(\rho{\to}\nu){\cap}\rho\} \vdash x : (\rho{\to}\nu){\cap}\rho}}{\{x{:}(\rho{\to}\nu){\cap}\rho\} \vdash x : \rho}}{\{x{:}(\rho{\to}\nu){\cap}\rho\} \vdash xx : \nu}}{\{z{:}\mu\} \vdash z\,\langle y = xx\rangle : \mu} \qquad \dfrac{\mathbf{D}}{\emptyset \vdash \lambda a.aa : (\sigma{\to}\tau){\cap}\sigma{\to}\tau}}{\{z{:}\mu\} \vdash z\,\langle y = xx\rangle\,\langle x = \lambda a.aa\rangle : \mu}$$

where **D** is the derivation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{a{:}(\sigma{\to}\tau){\cap}\sigma\} \vdash a : (\sigma{\to}\tau){\cap}\sigma}}{\{a{:}(\sigma{\to}\tau){\cap}\sigma\} \vdash a : \sigma{\to}\tau} \quad \dfrac{\dfrac{}{\{a{:}(\sigma{\to}\tau){\cap}\sigma\} \vdash a : (\sigma{\to}\tau){\cap}\sigma}}{\{a{:}(\sigma{\to}\tau){\cap}\sigma\} \vdash a : \sigma}}{\{a{:}(\sigma{\to}\tau){\cap}\sigma\} \vdash aa : \tau}}{\emptyset \vdash \lambda a.aa : ((\sigma{\to}\tau){\cap}\sigma){\to}\tau}$$

Fig. 3. A typing derivation

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.

One might note that, in Example 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 LC). 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 LC. But such a rule doesn't work: it is still too restrictive. For example, the reader can easily check that the term $z\langle y{=}xx\rangle\langle x{=}\lambda a.aa\rangle$ cannot be typed in such a system since $x \in \mathit{fv}(z\langle y{=}xx\rangle)$, 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 $z\langle y{=}xx\rangle\langle x{=}\lambda a.aa\rangle$ and this leads to the introduction of rule (K-cut).

Figure 3 shows how the term $z\langle y{=}xx\rangle\langle x{=}\lambda a.aa\rangle$ can be typed in system $\mathcal{E}$.

Notice that rule (cut) has no side-condition, and therefore, when $x \notin \mathit{av}(M)$ and $\Gamma \vdash N{:}\sigma$, one can freely use (cut) or (drop); when $x \notin \Gamma$ and $\Gamma \vdash N{:}\sigma$, one can freely use (cut) or (K-cut).

We now state some elementary properties of the type system, which highlight the relations between the non-standard cut rules.

Lemma 21 *(1) If $\Gamma' \leq \Gamma$, $\tau \leq \tau'$ and $\Gamma \vdash M\!:\!\tau$ then $\Gamma' \vdash M\!:\!\tau'$.*
(2) If $x \in \mathit{av}(M)$, then $\Gamma \vdash M\!:\!\tau$ implies $x \in \Gamma$.
(3) If $x \notin \mathit{av}(M)$, then $\Gamma \vdash M\!:\!\tau$ implies $\Gamma\backslash x \vdash M\!:\!\tau$.
(4) If $x \notin \mathit{av}(M)$, 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 (*4*) which follows immediately from parts (*1*) and (*3*).

- Before proving part (*1*) 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 (*1*), the only non-trivial case is when the last applied rule is $(\textsf{K-cut})$:

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

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 rule $(\textsf{K-cut})$. But $P^{y'}_y\langle y'\!=\!N\rangle$ is α-equivalent with $P\langle y\!=\!N\rangle$, so we are done.
- For part (*2*), three cases have to be looked at. The first one is when M is $P\langle y\!=\!N\rangle$ and the derivation ends with

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

Since $x \in \mathit{av}(M)$, by Lemma 4, 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 $x \in \mathit{av}(P)$ or $x \in \mathit{av}(N)$. In both cases the induction hypothesis yields $x \in \Gamma$. The other cases are

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

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

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

By Definition 20, the rules of system $\mathcal{E}$ are (start), $(\to\mathsf{I})$, $(\to\mathsf{E})$, $(\cap\mathsf{I})$, $(\cap\mathsf{E})$, (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 will write $\Gamma \vdash_{\mathcal{DLL}} M\!:\!\sigma$ if there exists a derivation with rules in $\mathcal{DLL}$ that has this as its conclusion, and similarly $\Gamma \vdash_{v\mathcal{BD}} M\!:\!\sigma$.

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 22 *(1) Rule* $(\mathsf{K\text{-}cut})$ *is derivable in system* $\mathcal{DLL}$*.*
(2) Rule (drop) *is derivable in system* $v\mathcal{BD}$*.*

PROOF.

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

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

can be replaced by an application of rule (drop), since, by Lemma 21(*2*), $y \notin \Gamma$ implies $y \notin \mathit{av}(P)$.

(2) : Consider an application of rule (drop):

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

By Lemma 21(*3*), $\Gamma\backslash y \vdash P\!:\!\tau$. Then the $(\mathsf{K\text{-}cut})$ rule yields $\Gamma\backslash y \vdash P\langle y\!=\!N\rangle\!:\!\tau$. Then, by Lemma 21(*4*), we have $\Gamma \vdash P\langle y\!=\!N\rangle\!:\!\tau$. ■

From the above Lemma we easily get:

Theorem 23 *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. We will use a generic notation for intersection types, $\sigma_1\cap\ldots\cap\sigma_n$, and assume that then each σ_i is not an intersection type.

Lemma 24 (Generation Lemma)

(1) $\Gamma \vdash x\!:\!\sigma$ *if and only if there exists* $(x{:}\tau) \in \Gamma$ *such that* $\tau \leq \sigma$.
(2) $\Gamma \vdash MN\!:\!\sigma$ *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* $\Gamma \vdash M\!:\!\tau_i \rightarrow \sigma_i$ *and* $\Gamma \vdash N\!:\!\tau_i$.
(3) $\Gamma \vdash \lambda x.M\!:\!\sigma$ *if and only if there exist* n*, and* ρ_i, τ_i $(i \in \underline{n})$ *such that* $\sigma \sim (\rho_1 \rightarrow \tau_1) \cap \ldots \cap (\rho_n \rightarrow \tau_n)$*, and* $\Gamma, (x{:}\rho_i) \vdash M\!:\!\tau_i$ *whenever* $i \in \underline{n}$.
(4) $\Gamma \vdash M\langle x\!=\!N\rangle\!:\!\sigma$ *if and only if either*
 (a) $x \in \mathit{av}(M)$*, and there exists* τ *such that* $\Gamma, (x{:}\tau) \vdash M\!:\!\sigma$ *and* $\Gamma \vdash N\!:\!\tau$*, or*
 (b) $x \notin \mathit{av}(M)$*,* $\Gamma \vdash M\!:\!\sigma$ *and there exist* Δ, τ *such that* $\Delta \vdash N\!:\!\tau$ *(in other words:* N *is typable in some environment).*

PROOF. The right-to-left implications immediately follow from the typing rules. The converse follows by easy induction on the structure of derivations. For part (*4*), notice that Theorem 23 allows us to skip the $(\mathsf{K\text{-}cut})$ rule. If the last applied rule is $(\cap\mathsf{I})$ we can use Lemma 21(*1*) and rule $(\cap\mathsf{I})$. ■

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

Theorem 25 (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 24(*2*), 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 $i \in \underline{n}$, $\Gamma \vdash \lambda x.M\!:\!\rho_i \rightarrow \sigma_i$ and $\Gamma \vdash N\!:\!\rho_i$. Then, by Lemma 24(*3*) $\Gamma, (x{:}\rho_i) \vdash M\!:\!\sigma_i$, and therefore, $\Gamma \vdash M\,\langle x\!=\!N\rangle\!:\!\sigma_i$ by rule (cut). So, by rule $(\cap\mathsf{I})$, $\Gamma \vdash M\,\langle x\!=\!N\rangle\!:\!\sigma$.

(App) : Then $\Gamma \vdash (MN)\langle x\!=\!P\rangle\!:\!\sigma$. Let $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$, then, by Lemma 24(*4*), we have two cases:

$(x \in \mathit{av}(MN)\ \&\ (\exists\,\tau).\Gamma, (x{:}\tau) \vdash MN\!:\!\sigma\ \&\ \Gamma \vdash P\!:\!\tau)$: Then we have $x \in \mathit{av}(M)$ or $x \in \mathit{av}(N)$, and, by Lemma 24(*2*), for every $i \in \underline{n}$, there exists ρ_i such that $\Gamma, (x{:}\tau) \vdash M\!:\!\rho_i \rightarrow \sigma_i$ and $\Gamma, (x{:}\tau) \vdash N\!:\!\rho_i$. Then $\Gamma \vdash M\,\langle x\!=\!P\rangle\!:\!\rho_i \rightarrow \sigma_i$ by rule (cut), and $\Gamma \vdash N\,\langle x\!=\!P\rangle\!:\!\rho_i$.

$(x \notin \mathit{av}(MN), \Gamma \vdash MN\!:\!\sigma\ \&\ (\exists\,\Delta, \tau).\Delta \vdash P\!:\!\tau)$: Then we have $x \notin \mathit{av}(M)$ and $x \notin \mathit{av}(N)$. As above, by Lemma 24(*2*), for every $i \in \underline{n}$, there exists ρ_i such that $\Gamma \vdash M\!:\!\rho_i \rightarrow \sigma_i$ and $\Gamma \vdash N\!:\!\rho_i$. Then, by rule (drop), $\Gamma \vdash M\,\langle x\!=\!P\rangle\!:\!\rho_i \rightarrow \sigma_i$ and also $\Gamma \vdash N\,\langle x\!=\!P\rangle\!:\!\rho_i$.

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

(Abs) : Then $\Gamma \vdash (\lambda y.M)\langle x = N\rangle : \sigma$. Let $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$. By Lemma 24(*4*), we have two cases:

$(x \in \textit{av}(M)$ & $(\exists \tau)\Gamma, (x{:}\tau) \vdash \lambda y.M : \sigma$ & $\Gamma \vdash N : \tau)$: By Lemma 24(*3*), for $i \in \underline{n}$, there exist ρ_i, μ_i such that $\sigma_i \sim \rho_i \rightarrow \mu_i$ and $\Gamma, (x{:}\tau), (y{:}\rho_i) \vdash M : \mu_i$. Then we get $\Gamma, (y{:}\rho_i) \vdash M\langle x = N\rangle : \mu_i$ by rule (cut).

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

In both cases, we obtain $\Gamma \vdash \lambda y.(M\langle x = N\rangle) : \sigma_i$ by rule $(\rightarrow\mathsf{I})$, and, by rule $(\cap\mathsf{I})$, also $\Gamma \vdash \lambda y.(M\langle x = N\rangle) : \sigma$.

(VarI) : Then $\Gamma \vdash x\langle x = N\rangle : \sigma$, and, by Lemma 24(*4*) $\Gamma, (x{:}\tau) \vdash x : \sigma$ and $\Gamma \vdash N : \tau$ for a certain τ. Then, by Lemma 24(*1*), $\tau \leq \sigma$, and, by Lemma 21(*1*), $\Gamma \vdash N : \sigma$.

(gc) : Then $\Gamma \vdash M\langle x = N\rangle : \sigma$ and $x \notin \textit{av}(M)$. Then, by Lemma 24(*4*), $\Gamma \vdash M : \sigma$. ■

Normal forms in λx are the same as in LC, and the type system $\mathcal{E}$ is an extension of the standard system of intersection types for LC. Therefore we get the typability of all normal forms for free. Moreover, we show that λ-free normal forms (that is to say, normal forms which are not λ-abstractions) have arbitrary types: this also holds in the the standard system of intersection types.

Lemma 26 (Normal forms are typable) *Let M be a normal form.*

(1) If M is λ-free and τ is a type, then there is an environment in which M has type τ.
(2) M is typable in some environment.

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 $\Gamma_1 \sqcap \ldots \Gamma_n \sqcap \{x{:}\tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \tau\} \vdash M : \tau$. 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, $\Gamma \backslash x \vdash M : \sigma \rightarrow \tau$. ■

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 27 (Subject Expansion) *If $M \rightsquigarrow N$ in one step, then*

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

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

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

PROOF.

(*1*) By induction on the structure 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 want to prove $\Gamma \vdash (PQ)\langle x{=}U\rangle : \sigma$. By Lemma 24(*2*), there are types τ_i, σ_i $(i \in \underline{n})$ such that $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$, and

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

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 24(*4*), which gives $\Gamma \vdash P : \tau_i{\rightarrow}\sigma_i$ 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

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

(which induces by rule (cut), $\Gamma \vdash (PQ)\langle x{=}U\rangle : \sigma_i$). In each case, we apply Lemma 24(*4*) on both P and Q.

- If $x \in$ *av*(P) and $x \notin$ *av*(Q), we get μ such that $\Gamma, (x{:}\mu) \vdash P : \tau_i{\rightarrow}\sigma_i$ and $\Gamma \vdash U : \mu$. Taking τ_i' to be μ, we use 21(*4*) 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 21(*4*) on P to get the result.
- If $x \in$ *av*(P) and $x \in$ *av*(Q), we get μ and ν, such that

$$\Gamma \vdash U : \mu, \ \ \&\ \ \Gamma \vdash U : \nu, \ \ \&\ \ \Gamma, (x{:}\mu) \vdash P : \tau_i{\rightarrow}\sigma_i \ \ \&\ \ \Gamma, (x{:}\nu) \vdash Q : \tau_i$$

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 24(*3*), we have types τ_i, σ_i $(i \in \underline{n})$ such that $\sigma \sim (\tau_1{\rightarrow}\sigma_1) \cap \ldots \cap (\tau_n{\rightarrow}\sigma_n)$ 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 $(\forall i \in \underline{n}).\Gamma \vdash (\lambda y.P)\langle x{=}U\rangle : \tau_i{\rightarrow}\sigma_i$. We apply Lemma 24(*4*) on $\Gamma, (y{:}\tau_i) \vdash P\langle x{=}U\rangle : \sigma_i$ and thereby,

- If $x \in \mathit{av}(P)$ we get μ such that $\Gamma, (y{:}\tau_i), (x{:}\mu) \vdash P : \sigma_i$ and $\Gamma, (y{:}\tau_i) \vdash U : \mu$. Since $y \notin \mathit{av}(U)$, applying Lemma 21(*1*) we get $\Gamma, (x{:}\mu), (y{:}\tau_i) \vdash P : \sigma_i$ and $\Gamma \vdash U : \mu$.
- If $x \notin \mathit{av}(P)$ we get that U is *typable* and $\Gamma, (y{:}\tau_i) \vdash P : \sigma_i$.

In both cases, we get the required result by applying first rule $(\rightarrow\mathsf{I})$ and then respectively rules (cut) or (drop).

(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 26, U is *typable*. We assume $\Gamma \vdash y : \sigma$, and 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.

(*2*) Again, the proof is by induction on the structure of 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 $x \in \mathit{av}(P)$, we have $(\exists\, \tau').\Gamma, (x{:}\tau') \vdash P : \tau \;\&\; \Gamma \vdash U : \tau'$, so $(\exists\, \tau').\Gamma \vdash \lambda x.P : \tau' \rightarrow \tau \;\&\; \Gamma \vdash U : \tau'$ which entails $\Gamma \vdash (\lambda x.P)U : \tau$ by rule $(\rightarrow\mathsf{E})$.
- If $x \notin \mathit{av}(P)$, then, using Lemma 24(*4*), we have $\Gamma \vdash P : \tau$ and $(\exists\, \Gamma', \tau').\Gamma' \vdash U : \tau'$. From Lemma 21(*1*), we get $\Gamma, (x{:}\tau') \vdash P : \tau$ which yields $\Gamma \vdash \lambda x.P : \tau' \rightarrow \tau$ by rule $(\rightarrow\mathsf{I})$. Hence $(\exists\, \Gamma', \tau').\Gamma \vdash \lambda x.P : \tau' \rightarrow \tau \;\&\; \Gamma' \vdash U : \tau'$. If we set Γ'' to be $\Gamma \sqcap \Gamma' \leq \Gamma$ we get $\Gamma'' \vdash \lambda x.P : \tau' \rightarrow \tau$ and $\Gamma'' \vdash U : \tau'$ which entails $\Gamma'' \vdash (\lambda x.P)U : \tau$.

$(M \equiv \lambda x.M')$: Then $N \equiv \lambda x.N'$, where $M' \rightsquigarrow N'$. We assume $\Gamma \vdash \lambda x.N' : \sigma$ and want to prove $\Gamma' \vdash \lambda x.M' : \sigma'$ for some environment $\Gamma' \leq \Gamma$ and type σ'. Using Lemma 24(3), 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 $\sigma' := \tau'_1 \rightarrow \sigma'_1$ we get $\Gamma' \vdash \lambda x.M' : \sigma'$ as required.

$(M \equiv M_1 M_2$ *where* M *is not its own perpetual redex*$)$: Then $N \equiv N_1 N_2$ where either $M_1 \rightsquigarrow N_1$ or M_1 is a λ-free normal form and $M_2 \rightsquigarrow N_2$ (see Definition 13). We assume $\Gamma \vdash N_1 N_2 : \sigma$, and want to prove $\Gamma' \vdash M_1 M_2 : \sigma$ for some environment $\Gamma' \leq \Gamma$. Using Lemma 24(*2*), we have types τ_i, σ_i $(i \in \underline{n})$ such that $\sigma \sim (\sigma_1 \cap \ldots \cap \sigma_n)$ and $(\forall\, i \in \underline{n}).\Gamma \vdash N_1 : \tau_i \rightarrow \sigma_i \;\&\; \Gamma \vdash N_2 : \tau_i$. Using Lemma 21(*1*) 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 Definition 13, 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 $\Gamma_i \vdash M_1 : \tau_i \rightarrow \sigma_i$, and using Lemma 21(*1*) 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 $\Gamma'_i \vdash M_2 : \tau'_i$. By Definition 13 we know that M_1 is a λ-free normal form, so Lemma 26(*1*) provides an environment Γ'' in which M_1 has type $\tau'_i \rightarrow \sigma_i$. Now, 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 Definition 13, 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 $M_2 \rightsquigarrow N_2$). Assume $\Gamma \vdash y\langle x{=}N_2\rangle : \sigma$. Using Lemma 24(*4*), we get $\Gamma \vdash y{:}\sigma$. Now by induction M_2 is *typable*. Hence applying 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 want to prove $\Gamma \vdash M_1\langle x{=}M_2\rangle : \sigma$.

$x \in \mathit{av}(N_1)$: Then, using Lemma 24(*4*), 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 $\Gamma \vdash M_1\langle x{=}M_2\rangle : \sigma$.

$x \notin \mathit{av}(N_1)$: Then using Lemma 21(*3*) we get $\Gamma\backslash x \vdash N_1\langle x{=}M_2\rangle : \sigma$. Then we can apply Lemma 24(*4*), 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 $(\mathsf{K\text{-}cut})$ and get $\Gamma \vdash M_1\langle x{=}M_2\rangle : \sigma$. ■

Corollary 28 (Weak Subject Expansion) *If $M \rightsquigarrow N$, then N is typable implies M is typable.*

Theorem 29 *All strongly normalizing terms are typable.*

PROOF. By induction on the length of the perpetual derivation. For the base case we observe that normal forms are typable (Lemma 26(*2*)), the induction step follows by Corollary 28. ■

6 All Typable Terms are Strongly Normalizable

The general idea of the reducibility method, is to interpret types by suitable sets (saturated and stable sets for Tait [40] and Krivine [27] and admissible relations for Mitchell [34,35]) 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 $\lambda\mathsf{x}$ and where the application is just the application of terms.

Definition 30 (Reducible terms)

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

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

(2) *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 31 *(1) $\mathcal{R} \subseteq \mathcal{SN}$.*
(2) $x\overrightarrow{N} \in \mathcal{SN} \Rightarrow \forall \rho\,[x\overrightarrow{N} \in \mathcal{R}^{\rho}]$.

PROOF. By simultaneous induction on the structure of types.

(1) (φ) : By Definition 30.

$(\sigma\to\tau)$: $M \in \mathcal{R}^{\sigma\to\tau} \Rightarrow (\mathit{IH}(2))\ M \in \mathcal{R}^{\sigma\to\tau}\ \&\ x \in \mathcal{R}^{\sigma} \Rightarrow (30)$
$Mx \in \mathcal{R}^{\tau} \Rightarrow (\mathit{IH}(1))\ Mx \in \mathcal{SN} \Rightarrow M \in \mathcal{SN}$.

$(\sigma\cap\tau)$: $M \in \mathcal{R}^{\sigma\cap\tau} \Rightarrow (30)\ M \in \mathcal{R}^{\sigma}\ \&\ M \in \mathcal{R}^{\tau} \Rightarrow (\mathit{IH}(1))\ M \in \mathcal{SN}$.

(2) (φ) : $x\overrightarrow{N} \in \mathcal{SN} \Rightarrow (30)\ x\overrightarrow{N} \in \mathcal{R}^{\varphi}$.

$(\sigma\to\tau)$:
$$\begin{aligned}
& x\overrightarrow{N} \in \mathcal{SN} && \Rightarrow (10, (\textsf{gen-var})) \\
& \forall M \in \mathcal{SN}\,[x\overrightarrow{N}M \in \mathcal{SN}] && \Rightarrow (\mathit{IH}(1)) \\
& \forall M \in \mathcal{R}^{\sigma}\,[x\overrightarrow{N}M \in \mathcal{SN}] && \Rightarrow (\mathit{IH}(2)) \\
& \forall M \in \mathcal{R}^{\sigma}\,[x\overrightarrow{N}M \in \mathcal{R}^{\tau}] && \Rightarrow (30)\ x\overrightarrow{N} \in \mathcal{R}^{\sigma\to\tau}
\end{aligned}$$

$(\sigma\cap\tau)$: $x\overrightarrow{N} \in \mathcal{SN} \Rightarrow (\mathit{IH}(2))\ x\overrightarrow{N} \in \mathcal{R}^{\sigma}\ \&\ x\overrightarrow{N} \in \mathcal{R}^{\tau} \Rightarrow (30)\ x\overrightarrow{N} \in \mathcal{R}^{\sigma\cap\tau}$. ■

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 33.

Lemma 32 (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 12). 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:

$$\begin{array}{lll}
(\sigma{\to}\tau): & (P\overrightarrow{\langle x=N\rangle}\langle y=Q\overrightarrow{\langle x=N\rangle}\rangle)\overrightarrow{M} \in \mathcal{R}^{\sigma\to\tau} & \Rightarrow (30) \\
& \forall R \in \mathcal{R}^{\sigma}\,[(P\overrightarrow{\langle x=N\rangle}\langle y=Q\overrightarrow{\langle x=N\rangle}\rangle)\overrightarrow{M}R \in \mathcal{R}^{\tau}] & \Rightarrow (\mathit{IH}) \\
& \forall R \in \mathcal{R}^{\sigma}\,[((P\langle y=Q\rangle)\overrightarrow{\langle x=N\rangle})\overrightarrow{M}R \in \mathcal{R}^{\tau}] & \Rightarrow (30) \\
& ((P\langle y=Q\rangle)\overrightarrow{\langle x=N\rangle})\overrightarrow{M} \in \mathcal{R}^{\sigma\to\tau}. &
\end{array}$$

$(\sigma{\cap}\tau)$: Immediate by Definition 30 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 33 (Soundness) *Suppose* $\{(x_1{:}\mu_1),\ldots,(x_n{:}\mu_n)\} \vdash M{:}\sigma$, *and, for* $i \in \underline{n}$, $N_i \in \mathcal{R}^{\mu_i}$, *with no* x_j *available in any* N_i. *Then* $M\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$.

PROOF. The proof is by induction on the structure of derivations. We will use the $\mathcal{SN}$-saturation of the saturated sets (Lemma 32) 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), $x_j\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$.

$({\to}$I$)$: Then $M \equiv \lambda y.M'$, $\sigma = \rho{\to}\tau$, 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 Definition 30, $\lambda y.M'\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\rho\to\tau}$. We can assume $y \notin \mathit{fv}(N)$, so, by rule (gen-Abs), $(\lambda y.M')\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\rho\to\tau}$.

$({\to}$E$)$: Then $M \equiv M_1M_2$ and there exists τ such that $\Gamma \vdash M_1{:}\tau{\to}\sigma$ and $\Gamma \vdash M_2{:}\tau$. By induction, $M_1\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\tau\to\sigma}$ and $M_2\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\tau}$. But then, by Definition 30, $M_1\overrightarrow{\langle x=N\rangle}M_2\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$, so, by rule (gen-App), $(M_1M_2)\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$.

$({\cap}$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 Definition 30, $M\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$.

$({\cap}$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 Definition 30, $M\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$.

(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 on the right-hand hypothesis, $Q\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\tau}$.

Then again by induction, but now 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 $P\,\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$. Since $y \notin \mathit{av}(P)$ we may use closure of $\mathcal{R}^{\sigma}$ under rule (gen-gc) to conclude that $(P\langle y=Q\rangle)\,\overrightarrow{\langle x=N\rangle} \in \mathcal{R}^{\sigma}$. To be able to apply that rule, we need that $Q \in \mathcal{SN}$; notice that by induction on the derivation for Q, $Q \in \mathcal{R}^{\tau}$, so, by Lemma 31(*1*), $Q \in \mathcal{SN}$.

(K-cut) : The proof is very similar to the (drop) case; we may also use to Theorem 23. ■

Theorem 34 *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 31(*2*), all term-variables are reducible for any type, so, by Theorem 33, for all M, $M\overrightarrow{\langle x=y\rangle}$ is reducible, where $\overrightarrow{y}$ are fresh. By Lemma 31(*1*) the term $M\overrightarrow{\langle x=y\rangle}$ is strongly normalizing, and since M is a subterm, the result follows. ■

7 Characterizing weak normalization and head normalization

The system $\mathcal{E}$ is obtained from the system $\mathcal{D}$ of [20] 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: this type was first added to intersection type assignment in [39]. The main feature of systems with intersection and ω is that typing is invariant under any conversion of subjects. In [20], characterizations of the head-normalizing and leftmost-normalizing terms of λx 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 λx, 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 rules (drop) and (K-cut) behave well in the presence of a universal type. In particular, we may ask whether the normalization theorems of [20] still hold in the presence of the new rules. In this section we show that this is the case. That is, we will verify that the $\mathcal{D}_{\omega}$-characterizations of normalizing and head-normalizing terms from [20] generalize in a 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 $\mathcal{D}_{\omega}$.

Definition 35 *The type system $\mathcal{E}_\omega$ is obtained from system $\mathcal{E}$ by adding the type constant ω and the rule:*

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

The type system $\mathcal{D}_\omega$ is obtained by adding ω and rule $(\omega\mathsf{I})$ to the system $\mathcal{D}$ of [20].

Theorem 36 *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 on the structure of 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

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

By induction we can derive $\Gamma \vdash M : \tau$ in $\mathcal{D}_\omega$, so certainly, using a $\mathcal{D}_\omega$-variant of Lemma 21(*4*), $\Gamma, (x{:}\omega) \vdash M : \tau$. By $(\omega\mathsf{I})$, $\Gamma \vdash N : \omega$ in $\mathcal{D}_\omega$, so we have

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

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

7.2 Head reduction and left-most reduction

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

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

Both these reductions could be considered a “head reduction.” In fact, it is our choice to consider them each to be head reductions.

Definition 37 (Head reduction) Head reduction *is the closure of the rules of $\lambda\mathrm{x}_{\mathsf{gc}}$ (Definition 5) under the structural rules of Figure 4. A term M is* head normalizing *if there is no infinite head-reduction starting from M. The set of head normalizing terms is denoted $\mathcal{H}N$.*

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

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

Fig. 4. Head reduction

$$\frac{M \xrightarrow{l} M' \quad M \text{ not an abstraction}}{MN \xrightarrow{l} M'N} \qquad \frac{M \xrightarrow{l} M' \quad M \text{ not an abstraction}}{M\langle x{=}N\rangle \xrightarrow{l} M'\langle x{=}N\rangle}$$

$$\frac{M \xrightarrow{l} M'}{\lambda x.M \xrightarrow{l} \lambda x.M'} \qquad \frac{M_i \xrightarrow{l} M_i' \quad M_i \text{ left-most non-normal}}{xM_1 ... M_i ... M_n \xrightarrow{l} xM_1 ... M_i' ... M_n}$$

Fig. 5. Left-most reduction

Definition 38 (Left-most reduction) Left-most reduction *is the closure of the rules of $\lambda\mathtt{x}_{\mathsf{gc}}$ under the structural rules in Figure 5. 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 $\mathcal{L}N$.*

Observe that, in contrast to the classical notions, both head reduction and left-most reduction are non-deterministic strategies. Indeed, each of the 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 to $Q' \equiv M\langle y{=}L\rangle\langle x{=}\ N\langle y{=}L\rangle\rangle$.

7.3 Characterization theorems

We will assume familiarity with [20] 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 [20]. In this section we refer to the traditional garbage collection rule as gc^-:

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

Formally, since [20] 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 [13]: 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:

(1) ω is trivial,
(2) If σ is trivial and τ is any type, then $\tau{\rightarrow}\sigma$ is trivial,
(3) If σ and τ are trivial, then $\sigma \cap \tau$ is trivial.

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

Lemma 39 *If M is typable with a non-trivial type in system $\mathcal{D}_\omega$ then M is head-normalizing in the calculus $\lambda\mathtt{x}_{\mathsf{gc}}$.*
If M is typable in system $\mathcal{D}_\omega$ with a type not involving ω then M is left-most-normalizing in the calculus $\lambda\mathtt{x}_{\mathsf{gc}}$.

PROOF. Each of these assertions is proved in [20] 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 [20] is unchanged, completing the proof. ■

Theorem 40 *Let M be a closed term. The following are equivalent.*

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

PROOF. By Theorem 36 we may replace, in (*1*), "$\mathcal{E}_\omega$" by "$\mathcal{D}_\omega$." Then each of the equivalences has been proved in [20] with the exception of the implication from (*1*) to (*2*) since, in [20] garbage collection refers to the more restricted rule gc^-. But for this implication we use Lemma 39 here. ■

Theorem 41 *Let M be a closed term. The following are equivalent.*

(1) M is typable in system $\mathcal{E}_\omega$ with a type not involving ω.
(2) M is typable with a proper type in system $\mathcal{E}_\omega$.
(3) M is left-most-normalizing in the calculus $\lambda\mathtt{x}_{\mathsf{gc}}$.
(4) M is left-most-normalizing in the calculus $\lambda\mathtt{x}$ (without garbage collection).
(5) M has a normal form.

PROOF. As for Theorem 40. ■

In Theorem 41, the implications 5 to 3 and 5 to 4 state that in $\lambda\mathtt{x}$ and $\lambda\mathtt{x}_{\mathsf{gc}}$ 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 note 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 [11,20]. 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, Frédéric Lang, Simona Ronchi della Rocca, and Kristoffer Rose for many helpful discussions. Moreover they thank the referees of the conferences LATIN'02 and TCS'02, and of the present submission 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] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102(1):135–163, 1992.

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

[5] 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.

[6] 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), 1984. Second edition.

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

[8] 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.

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

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

[11] 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.

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

[13] 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, 1990.

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

[15] 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.

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

[17] 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.

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

[19] 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.

[20] D. Dougherty and P. Lescanne. Reductions, intersection types, and explicit substitutions. *Mathematical Structures in Computer Science*, 13(1):55–85, 2003.

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

[22] J. Gallier. Typing untyped lambda terms, or reducibility strikes again. *Annals of Pure and Applied Logic*, 91:231–270, 1998.

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

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

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

[26] 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.

[27] J.-L. Krivine. *Lambda-calcul, types et modèles*. Masson, 1990.

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

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

[30] S. Lengrand, D. Dougherty, and P. Lescanne. An improved system of intersection types for explicit substitutions. In R. A. Baeza-Yates, U. Montanari and N. Santoro, editors, *Conference on Theoretical Computer Science, IFIP Congress*, pages 511–524. Kluwer Academic Publishers, 2002.

[31] 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.

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

[33] 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.

[34] 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), 1990.

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

[36] 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.

[37] 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.

[38] 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.

[39] P. Sall´e. Une extension de la th´eorie des types en λ-calcul. In G. Ausiello and C. B¨ohm, editors, *ICALP'78*, volume 62 of *Lecture Notes in Computer Science*, pages 398–410. Springer-Verlag, 1978.

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