Math. Struct. in Comp. Science (2001), *vol.* 11, *pp.* 47–90. Printed in the United Kingdom
© 2001 Cambridge University Press

Perpetuality in a named lambda calculus with explicit substitutions

EDUARDO BONELLI

Laboratoire de Recherche en Informatique, Bât 490, Université de Paris-Sud, 91405, Orsay Cedex, France.

and

Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina.

Email: bonelli@lri.fr

Received 30 September 1999; revised 15 April 2000

We study perpetuality in the calculus of explicit substitutions $\lambda\mathtt{x}$. A reduction is called *perpetual* if it preserves the possibility of infinite reduction sequences. We then take a look at applications of this study: an inductive characterization of the $\lambda\mathtt{x}$-strongly normalizing terms, two perpetual reduction strategies for $\lambda\mathtt{x}$ and finally a proof of strong normalization of a polymorphic lambda calculus with explicit substitutions F_{es}. To complete the study of F_{es}, the property of subject reduction is shown to hold by extending type assignments of the typing rules to allow non-pure types (types with possible occurrences of the type substitution operator).

1. Introduction

Substitution is a fundamental operation in the lambda calculus. From a computational cost point of view, in terms of atomic reduction steps implementing this operation, the picture is not so clear. The β-reduction axiom $(\lambda x.M)N \to_\beta M\{x \leftarrow N\}$ may be interpreted as: the result of executing function $\lambda x.M$ over argument N (the actual parameter) is obtained by substituting N for all (free) occurrences of x (the formal parameter) in M. But any implementation of this axiom must include instructions for computing this process. Although at the meta-level an execution of the β-reduction axiom is atomic, the cost of computing $M\{x \leftarrow N\}$ is highly dependent on the form of M, and even more so if unwanted variable capture conflicts must be avoided by renaming bound variables. Calculi of explicit substitution incorporate the substitution operation into the calculus and cater for its behaviour by incorporating additional reduction rules. Thus the study of the (non-trivial) process of substitution may be developed in a uniform manner. In this way, calculi of explicit substitution are said to eliminate the gap between theory and practice in the implementation of functional programming languages.

The study of explicit substitution calculi was pioneered by the $\lambda\sigma$-calculus (Abadi *et*

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

al. 1991), formulated in a de Bruijn indices setting. Problems such as confluence (on open terms) and preservation of strong normalization suffered by $\lambda\sigma$ motivated further studies in the area: other calculi formulated are $\lambda\sigma_{\Uparrow}$ (Hardin and Lévy 1989), $\lambda\upsilon$ (Lescanne 1994), λs (Kamareddine and Ríos 1995), λd (Ferreira *et al.* 1996), $\lambda\zeta$ (Muñoz 1996a) and $\lambda\mathtt{x}$ (Rose 1992). All but the last of these calculi have been formulated with de Bruijn indices; $\lambda\mathtt{x}$ uses variable names.

Perpetuality studies the contraction of redexes that preserve the possibility of infinite reduction sequences. Perpetual reduction strategies for the λ-calculus were introduced in Barendregt *et al.* (1976). An interesting survey on perpetual reductions in the λ-calculus is van Raamsdonk *et al.* (1999). As regards uniformly normalizable calculi (all redexes are perpetual), Klop (Klop 1980) showed that all non-erasing orthogonal CRSs (hence all non-erasing orthogonal Term Rewriting Systems) enjoy this property, thus generalizing Church's Theorem (Church and Rosser 1936) stating that the λ_I-calculus (for every abstraction $\lambda x.M$ there is at least one free variable occurrence of x in M) is uniformly normalizable. In the direction of characterizing perpetual redexes, Khasidashvili (Khasidashvili 1994) proved that all non-erasing redexes are perpetual in orthogonal ERSs (hence all non-erasing redexes are perpetual in orthogonal Term Rewriting Systems), thus generalizing the Conservation Theorem (Barendregt *et al.* 1976) stating that β_I-redexes are perpetual in the λ-calculus. All these results are formulated for orthogonal systems, but the calculus of explicit substitutions $\lambda\mathtt{x}$ is not orthogonal, and hence the results on perpetuality already developed do not apply. So we study perpetuality in the $\lambda\mathtt{x}$-calculus by adapting a technique originally introduced for proving preservation of strong normalization of calculi of explicit substitutions (Benaissa *et al.* 1996). Applications of this study are a formulation of an inductive definition of the set of $\lambda\mathtt{x}$-strongly normalizable terms, two perpetual reduction strategies for $\lambda\mathtt{x}$ (based on van Raamsdonk and Severi (1995)), of which one is effective, and a proof of strong normalization of typed terms in a polymorphic lambda-calculus with explicit substitutions.

The polymorphic lambda calculus, called the F-calculus, was introduced by J-Y.Girard (Girard 1972) and independently rediscovered by J.C.Reynolds (Reynolds 1974). It consists of extending the simply typed λ-calculus with parametric polymorphism. Its interest from a programming point of view is that although all typable terms are terminating, the computable functions that may be expressed are not restricted to the primitive recursive functions but also include, for example, Ackermann's function. We will study an explicit substitution formulation of F based on $\lambda\mathtt{x}$.

As regards related work, Bloo defines Explicit Pure Type Systems (EPTS) in his thesis (Bloo 1997). In this formalism the Pure Type Systems (PTS) (Barendregt 1992) are extended with explicit substitutions in a named variable setting. He proves that $\mathscr{R}$ is strongly normalizing if and only if $\mathscr{R}\mathtt{x}$ is strongly normalizing, where $\mathscr{R}$ is a PTS and $\mathscr{R}\mathtt{x}$ is its explicit substitution version in the EPTS formalism, each with its corresponding notion of reduction. On the other hand, failure of subject reduction for $F\mathtt{x}$ (that is, the explicit substitution version of the polymorphic lambda calculus F) is illustrated by providing counter-examples (and for other systems too). We shall remedy this situation: the subject reduction property will be shown to hold for F_{es}. Also, our presentation differs substantially in that our proof of strong normalization relies on the reducibility

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

technique, and thus does not require the intermediate result of strong normalization of F. Independently, in Bloo (1999), the notion of Explicit Type System is introduced – this presentation of pure type systems with explicit substitutions enjoys the subject reduction property. It is also worth mentioning that our calculus uses two operators of explicit substitutions: one for terms and one for types. Since one of the primary goals leading to the introduction of explicit substitution operators is to 'implement' the notion of meta-level substitution, which has the benefit that unnecessary propagation of substitutions is avoided in that substitutions for terms do not propagate to types.

Finally, this study is related to the author's paper Bonelli (1999), which is, essentially, the work reported in Section 5 on the polymorphic lambda calculus with explicit substitutions F_{es}. The present work deals with perpetuality in the calculus of explicit substitutions $\lambda\mathtt{x}$ and shows, as an application, that the proof of strong normalization (SN) of F_{es} (Bonelli 1999) can be greatly simplified. Indeed, in Bonelli (1999) in order for the proof of SN of F_{es} to go through, the following characterization of the $\lambda\mathtt{x}$-strongly normalizing terms (denoted $SN_{\lambda\mathtt{x}}$) (Bloo and Geuvers 1998) was required (see Section 2 for notation):

$$SN_{\lambda\mathtt{x}} = \{M \in \mathscr{T}_{\lambda\mathtt{x}} | \forall\ N \subseteq M,\ \mathtt{x}(N) \in SN_{\beta}\}.$$

In the present work we drop the need for that characterization. The perpetuality analysis has allowed us to obtain a 'pure' reducibility proof.

The structure of the rest of this paper is as follows: Section 2 recalls very briefly some basic notions of λ-calculus and the $\lambda\mathtt{x}$-calculus. Section 3 studies infinite reductions in $\lambda\mathtt{x}$. Section 4 introduces some applications of the perpetuality analysis: an inductive characterization of the $\lambda\mathtt{x}$-strongly normalizing terms, two perpetual reduction strategies for $\lambda\mathtt{x}$, and, finally, strong normalization of F_{es} is mentioned, deferring all details to Section 5.3. Section 5 presents the untyped F_{es}-calculus and the typing rules for typed F_{es}, and then concentrates on the subject reduction property and the proof of F_{es}-strong normalization of all typable terms. Finally, we present our conclusions and suggest future research directions.

2. Lambda calculus and explicit substitutions

In this section we briefly introduce the (untyped) λ-calculus (Barendregt 1984) and the calculus of explicit substitutions $\lambda\mathtt{x}$ (Rose 1992) in order to fix notation. Let $\mathscr{X}$ be a denumerably infinite set of variables. Then the set of terms of the lambda calculus, denoted $\mathscr{T}_{\lambda}$, is defined by the following grammar: $M ::= x \mid MM \mid \lambda x.M$ where $x \in \mathscr{X}$. We use letters $x, y, z, \ldots$ for variables and $M, N, O, \ldots$ for terms. The notion of free and bound variables are defined as usual, and the following variable convention is adopted: names of bound variables are different from the names of free variables, and, moreover, different occurrences of the abstraction operator have different binding variables. A precise formulation of α-conversion will be given in Section 5. The set of free variable of a term M is denoted $FV(M)$. The β-rule is stated as $(\lambda x.M)N \to_{\beta} M\{x \leftarrow N\}$. β-reduction is the contextual closure of the β-rule. The expression $M\{x \leftarrow N\}$ denotes the term resulting from substituting N for every free occurrence of the variable x in M. This operation takes place at the *meta-level*, thus it is external to the calculus.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

A *position* in a term is defined as usual: letters p, q are used for positions and $\prec$ for the corresponding prefix ordering. A *context* is a term with one hole: letters $C, D, \ldots$ are used for contexts. If C is a context with the hole at position p and M is a term, we write $C\{M\}_p$ for the term resulting from the replacement of the term M for the hole.

A term M is said to be β-*strongly normalizing* if all β-reduction sequences starting from M terminate. The set of terms of the λ-calculus that are β-strongly normalizing will be denoted SN_β. Also, we shall use the symbol $\equiv$ to denote syntactical equality modulo α-conversion (sometimes we shall use $\equiv_{L\ n}$ when the result follows by applying Lemma n) and $\overset{*}{\rightarrow}_\beta$ for the reflexive and transitive closure of $\rightarrow_\beta$. For the subterm relation we use $\subseteq$, that is, $M \subseteq N$ is read as 'M is a subterm of N', and $M \subset N$ is read as 'M is a proper subterm of N'. Two well-known properties (Barendregt 1984) are given by the following lemmas.

Lemma 2.1 If $M \rightarrow_\beta M'$, then for every term N we have:

1 $M\{x \leftarrow N\} \rightarrow_\beta M'\{x \leftarrow N\}$.

2 $N\{x \leftarrow M\} \overset{*}{\rightarrow}_\beta N\{x \leftarrow M'\}$.

Lemma 2.2 (Substitution lemma) $M\{x \leftarrow N\}\{y \leftarrow O\} \equiv M\{y \leftarrow O\}\{x \leftarrow N\{y \leftarrow O\}\}$ if $x \notin FV(O)$.

The λx-calculus (Rose 1992; Bloo 1997) is a calculus of explicit substitutions for the λ-calculus formulated in a named variable setting. The operation of substitution is incorportated at the object-level, thus the β-reduction rule is transformed into the following *Beta*-reduction rule:

$$(\lambda x.M)N \ \rightarrow_{Beta} \ M\langle x := N\rangle$$

where the operator $.\langle . := .\rangle$ is a new operator in the calculus called a *closure* or *substitution operator*. Thus rules describing its behaviour must be introduced.

Given a denumerably infinite set of variables $\mathscr{X}$, the set of terms of λx, denoted $\mathscr{T}_{\lambda\mathrm{x}}$, is defined by the grammar: $M ::= x \mid MM \mid \lambda x.M \mid M\langle x := M\rangle$ where $x \in \mathscr{X}$. The subset $\mathscr{T}_\lambda \subset \mathscr{T}_{\lambda\mathrm{x}}$ are called *pure terms*. The aforementioned variable convention is maintained, with the additional observation that the variable x in the term $M_1\langle x := M_2\rangle$ binds the free occurrences of x in M_1. The rules of the λx-calculus are

$$\begin{array}{lll}
(\lambda x.M)N & \rightarrow_{Beta} & M\langle x := N\rangle \\
(MN)\langle x := P\rangle & \rightarrow_{app} & M\langle x := P\rangle N\langle x := P\rangle \\
(\lambda y.M)\langle x := P\rangle & \rightarrow_{lam} & \lambda y.M\langle x := P\rangle \\
x\langle x := P\rangle & \rightarrow_{var} & P \\
M\langle x := P\rangle & \rightarrow_{gc} & M \qquad\qquad x \notin FV(M)
\end{array}$$

The λx-calculus without the *Beta*-rule is called the *substitution subcalculus of* λx and is denoted by x. This subcalculus is strongly normalizing and confluent, and its normal forms are pure terms (Bloo 1997, Chapter 13). Thus if $M \in \mathscr{T}_{\lambda\mathrm{x}}$, we use $\mathrm{x}(M)$ to denote its unique x-normal form. The *gc*-rule is referred to as the *garbage collection rule*: the substitution body P in this rule is called *garbage*. A variant of the λx-calculus is the $\lambda\mathrm{x}^-$-calculus whose rules are those of the λx-calculus except for the *gc*-rule, which is replaced

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

by the more *restricted garbage collection rule* $y\langle x := P\rangle \to_{rgc} y$ where $x \not\equiv y$. Since $\lambda\mathtt{x}$ is more general than $\lambda\mathtt{x}^-$ (in the sense that $\to_{\lambda\mathtt{x}^-} \subset \to_{\lambda\mathtt{x}}$ but $\to_{\lambda\mathtt{x}} \not\subset \to_{\lambda\mathtt{x}^-}$), we shall mainly deal with the $\lambda\mathtt{x}$-calculus. Nevertheless, the main results on perpetuality established in this work also hold for the $\lambda\mathtt{x}^-$-calculus, as expected. In fact, many proofs are simpler when the $\lambda\mathtt{x}^-$-calculus is considered.

Remark 2.3 Every $\lambda\mathtt{x}^-$-redex is a $\lambda\mathtt{x}$-redex.

As for β-reduction, we use $\to_{\lambda\mathtt{x}}$ for one-step reduction, $\overset{+}{\to}_{\lambda\mathtt{x}}$ for its transitive closure, $\overset{*}{\to}_{\lambda\mathtt{x}}$ for its reflexive transitive closure, and, additionally, we add a position where the reduction takes place if it is necessary (for example, $\to_{\lambda\mathtt{x},p}$, where ϵ is used for the root position). If M has an infinite $\lambda\mathtt{x}$-reduction sequence, we write $\infty(M)$. Also, if $M, N_1, \ldots, N_m \in \mathscr{T}_{\lambda\mathtt{x}}$ and $x_1, \ldots, x_m \in \mathscr{X}$ for $m \geqslant 0$, we use $M^\star$ to denote the term $M\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle$. We use $SN_{\lambda\mathtt{x}}$ to denote the set of $\lambda\mathtt{x}$-strongly normalizing terms in $\mathscr{T}_{\lambda\mathtt{x}}$; $|M|$ denotes M's size (that is, the number of variables, applications, abstractions and closures) and if $M \in SN_{\lambda\mathtt{x}}$, then $maxred(M)$ denotes the length of the longest reduction sequence starting from M to its (unique) $\lambda\mathtt{x}$-normal form. Note that when $M \in SN_{\lambda\mathtt{x}}$ we have $maxred(M)$ is well-defined.

The following result shows that the substitution subcalculus $\mathtt{x}$ correctly implements meta-level substitution.

Lemma 2.4 $\mathtt{x}(M\langle y := N\rangle) \equiv \mathtt{x}(M)\{y \leftarrow \mathtt{x}(N)\}$.

3. Perpetuality in $\lambda\mathtt{x}$

This section studies infinite reductions in $\lambda\mathtt{x}$ by making use of the closure tracing technique introduced in Benaissa *et al.* (1996) for proving preservation of strong normalization (PSN) for the calculus $\lambda\upsilon$. In Benaissa *et al.* (1996) infinite derivations starting from pure terms are studied, here we study infinite derivations starting from *any* term. Also, some minor simplifications on this technique, as a tool for proving PSN, are suggested.

Note that since $\lambda\mathtt{x}$ is erasing and non-orthogonal (in the sense that there are non-trivial critical pairs, such as for example, the one generated by the overlap between the rules *Beta* and *app*) the results on perpetuality already developed for orthogonal non-erasing reduction systems do not apply (see, for example, Klop (1980), Klop (1992), Khasidashvili *et al.* (1999) and Khasidashvili (1994)).

Definition 3.1 A substitution $\langle x := N\rangle$ is called *void in* $M\langle x := N\rangle$ if $x \notin FV(\mathtt{x}(M))$; M is said to be the *target* of the substitution and N its *body*. *Void reduction*, denoted $\overset{v}{\to}_{\lambda\mathtt{x}}$, is $\lambda\mathtt{x}$-reduction inside the body of a void substitution (more precisely, it is the contextual closure of the reduction: $M\langle z := N\rangle \to_{\lambda\mathtt{x}} M\langle z := N'\rangle$ if $N \to_{\lambda\mathtt{x}} N'$ and $z \notin FV(\mathtt{x}(M))$).

The following two lemmas together are taken from Bloo (1997, Lemma 3.15 and Proposition 4.15).

Lemma 3.1 (Projection) Let M, N be terms in $\mathscr{T}_{\lambda\mathtt{x}}$. Then:

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

1 If $M \to_{\lambda \mathtt{x}} N$, then $\mathtt{x}(M) \stackrel{*}{\to}_{\beta} \mathtt{x}(N)$.
2 If $M \to_{Beta} N$ is not a void reduction, then $\mathtt{x}(M) \stackrel{+}{\to}_{\beta} \mathtt{x}(N)$.

Lemma 3.2 If $M_0, M_1, \ldots \in \mathscr{T}_{\lambda \mathtt{x}}$ such that $\mathtt{x}(M_0) \in SN_{\beta}$ and $M_0 \to_{\lambda \mathtt{x}} M_1 \to_{\lambda \mathtt{x}} \ldots$ is an infinite λx-reduction sequence, there is a $k \in \mathbb{N}$ such that for all $i \geqslant k$, $M_i \stackrel{v}{\to}_{\lambda \mathtt{x}} M_{i+1}$.

Proof. Since $\to_{\mathtt{x}}$ is strongly normalizing, we may assume that the infinite λx-reduction sequence has the form $M_0 \stackrel{*}{\to}_{\mathtt{x}} M_1 \to_{Beta} M_2 \stackrel{*}{\to}_{\mathtt{x}} M_3 \ldots$. Now, by Lemma 3.1(1), we have $\mathtt{x}(M_0) \stackrel{*}{\to}_{\beta} \mathtt{x}(M_2) \stackrel{*}{\to}_{\beta} \mathtt{x}(M_4) \stackrel{*}{\to}_{\beta} \mathtt{x}(M_6) \ldots$, where by Lemma 3.1(2) we have $\mathtt{x}(M_{2n}) \stackrel{+}{\to}_{\beta} \mathtt{x}(M_{2n+2})$ if $M_{2n+1} \to_{Beta} M_{2n+2}$ is not void.

Now, since $\mathtt{x}(M_0) \in SN_{\beta}$, there is a $j \in \mathbb{N}$ such that for all $i \geqslant j$ we have $M_{2i+1} \stackrel{v}{\to}_{Beta} M_{2i+2}$. We must now prove that from some point onwards not only the $\to_{Beta}$ reductions are void but also the $\to_{\mathtt{x}}$ reductions. This is done by defining an interpretation h on $\mathscr{T}_{\lambda \mathtt{x}}$:

$$
\begin{array}{lll}
h(x) & =_{def} & 1 \\
h(MN) & =_{def} & h(M) + h(N) + 1 \\
h(\lambda z.N) & =_{def} & h(N) + 1 \\
h(M\langle z := N\rangle) & =_{def} & \begin{cases} (1 + h(N)) \times h(M) & \text{if } z \in FV(\mathtt{x}(M)) \\ 2 \times h(M) & \text{if } z \notin FV(\mathtt{x}(M)). \end{cases}
\end{array}
$$

One may then verify that:

— if $M \stackrel{v}{\to}_{\lambda \mathtt{x}} N$, then $h(M) = h(N)$, and
— if $M \to_{\mathtt{x}} N$ is not void, then $h(M) > h(N)$.

Thus there must be a $k > j$ such that for all $i \geqslant k$ we have that not only $M_{2i+1} \stackrel{v}{\to}_{Beta} M_{2i+2}$ but also $M_{2i} \stackrel{v}{\to}_{\mathtt{x}} M_{2i+1}$. □

Note that in contrast to Benaissa *et al.* (1996) there has been no need to define internal/external positions and to prove commutation of internal/external reductions in order to prove Lemma 3.2 (Benaissa *et al.* 1996, Lemma 13). This is clearly an advantage since commutation results are long, tedious and technical. It thus suggests a simplified variant of the proof technique introduced in Benaissa *et al.* (1996) for proving PSN.

Lemma 3.3 (One-step closure tracing) Let $M, N \in \mathscr{T}_{\lambda \mathtt{x}}$ with $M \to_{\lambda \mathtt{x}} N \equiv C\{P\langle y := Q\rangle\}_p$. Then,

1 either $M \equiv C'\{P'\langle y := Q\rangle\}_{p'}$,
2 or, $M \equiv C'\{P'\langle y := Q'\rangle\}_{p'}$ and $Q' \to_{\lambda \mathtt{x}} Q$,
3 or, $M \equiv C\{(\lambda y.P)Q\}_p$.

Proof. We consider the following two cases:

— The reduction takes place at the root. First note that if $P\langle y := Q\rangle$ occurs in a subterm of N that is also a subterm of M, then for some context C', $P' \equiv P$ and $Q' \equiv Q$ the first item holds trivially. Also note that this includes the cases where the rule applied at the root is *var* or *gc*. Otherwise we must have one of the following:
  – $M \equiv (\lambda y.P)Q \to_{Beta} P\langle y := Q\rangle \equiv N$ with $C \equiv \Box$. Then the third item holds.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

- $M \equiv (M_1 M_2)\langle y := Q\rangle \rightarrow_{app} M_1\langle y := Q\rangle M_2\langle y := Q\rangle \equiv N$ where P is M_1 or M_2. Then we take $P' = M_1 M_2$, $C' \equiv \Box$ and $Q' \equiv Q$ and the first item holds.
- $M \equiv (\lambda z.P)\langle y := Q\rangle \rightarrow_{lam} \lambda z.P\langle y := Q\rangle \equiv N$. Then we take $C' \equiv \Box$, $P' \equiv \lambda z.P$ and $Q' \equiv Q$ and item one holds.

— The reduction is internal. We use induction on M.

- $M \equiv x$. The result holds vacuously.
- $M \equiv M_1 M_2$ with either $M_1 \rightarrow_{\lambda x} M_1'$, or $M_2 \rightarrow_{\lambda x} M_2'$. We consider the first case, the other being similar. Thus $N \equiv M_1' M_2$. We have two further cases to consider.
  - The subterm $P\langle y := Q\rangle$ occurs in M_1'. We then use the induction hypothesis.
  - The subterm $P\langle y := Q\rangle$ occurs in M_2. Then the first item trivially holds.
- $M \equiv \lambda z.M_1$ with $M_1 \rightarrow_{\lambda x} M_1'$. We use the induction hypothesis.
- $M \equiv M_1\langle z := M_2\rangle$ with
  - either, $M_1 \rightarrow_{\lambda x} M_1'$ and $N \equiv M_1'\langle z := M_2\rangle$. Then if $P\langle y := Q\rangle$ occurs in M_1', we use the induction hypothesis. If it occurs in M_2, the first item trivially holds. Finally, if $N \equiv P\langle y := Q\rangle$, we take $C' \equiv \Box$, $P' \equiv M_1$ and $Q' \equiv M_2$ and the first item holds.
  - or, $M_2 \rightarrow_{\lambda x} M_2'$ and $N \equiv M_1\langle z := M_2'\rangle$. Then if $P\langle y := Q\rangle$ occurs in M_1, the first item trivially holds. If it occurs in M_2', we use the induction hypothesis. Finally, if $N \equiv P\langle y := Q\rangle$, we take $C' \equiv \Box$, $P' \equiv M_1$ and $Q' \equiv M_2$, and the second item holds. □

This result extends naturally to many-step reduction.

Lemma 3.4 (Closure tracing) Let $M_1, \ldots, M_n \in \mathcal{T}_{\lambda x}$ such that $M_i \rightarrow_{\lambda x} M_{i+1}$ for $i \in 1..n-1$, and $M_n \equiv C\{P\langle y := Q\rangle\}_p$. Then,

1 either there is an $i \in 1..n-1$ such that $M_i \equiv C'\{(\lambda y.P')Q'\}_{p'}$ and $Q' \overset{*}{\rightarrow}_{\lambda x} Q$,
2 or, $M_1 \equiv C'\{P'\langle y := Q'\rangle\}_{p'}$ and $Q' \overset{*}{\rightarrow}_{\lambda x} Q$.

Proof. The proof is by induction on n using Lemma 3.3. □

The following notion of skeleton first appeared in Kamareddine and Ríos (1995). The corresponding concept in terms of positions occurs under the name Frontier (Benaissa *et al.* 1996).

Definition 3.2 (Skeleton) The *skeleton* of a term M in $\mathcal{T}_{\lambda x}$, denoted $\mathcal{SK}(M)$, is defined inductively as follows:

$$\begin{array}{lll} \mathcal{SK}(x) & =_{def} & x \\ \mathcal{SK}(N_1 N_2) & =_{def} & \mathcal{SK}(N_1)\mathcal{SK}(N_2) \\ \mathcal{SK}(\lambda y.N) & =_{def} & \lambda y.\mathcal{SK}(N) \\ \mathcal{SK}(N_1\langle z := N_2\rangle) & =_{def} & \mathcal{SK}(N_1)\langle z := .\rangle \end{array}$$

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

where the '.' symbol may be seen as a place-holder (much the same as the '□' symbol in a context). Intuitively, the skeleton of a term is the part where only reductions that do not take place in bodies of substitutions are possible (these are sometimes also called external reductions).

Remark 3.5 Note that if $M \overset{v}{\to}_{\lambda \mathtt{x}} N$, then $\mathscr{SK}(M) \equiv \mathscr{SK}(N)$.

We can say more on infinite reductions in $\lambda\mathtt{x}$.

Lemma 3.6 If $M_1, M_2, \ldots \in \mathscr{T}_{\lambda\mathtt{x}}$ such that $\mathtt{x}(M_1) \in SN_\beta$ and $M_1 \to_{\lambda\mathtt{x}} M_2 \to_{\lambda\mathtt{x}} \ldots$ is an infinite $\lambda\mathtt{x}$-reduction sequence, then there exists $k \in \mathbb{N}$, a position p, a variable y, a context C and $\lambda\mathtt{x}$ terms $P, Q_k, Q_{k+1}, Q_{k+2}, \ldots$ such that

$$\begin{array}{lll} M_1 \to_{\lambda\mathtt{x}} M_2 \to_{\lambda\mathtt{x}} \ldots \to_{\lambda\mathtt{x}} M_k & \equiv & C\{P\langle y := Q_k\rangle\}_p \\ & \overset{v}{\to}_{\lambda\mathtt{x}} & C\{P\langle y := Q_{k+1}\rangle\}_p \\ & \overset{v}{\to}_{\lambda\mathtt{x}} & C\{P\langle y := Q_{k+2}\rangle\}_p \\ & & . \\ & & . \end{array}$$

Proof. Consider an infinite $\lambda\mathtt{x}$-reduction sequence $\mathscr{D} : M \equiv M_1 \to_{\lambda\mathtt{x}} M_2 \to_{\lambda\mathtt{x}} M_3 \ldots$ starting from M. Then, since $\mathtt{x}(M) \in SN_\beta$, we may apply Lemma 3.2 and obtain a $k \in \mathbb{N}$ such that for all $i \geqslant k$ we have $M_i \overset{v}{\to}_{\lambda\mathtt{x}} M_{i+1}$. Hence, by Remark 3.5, we have $\mathscr{SK}(M_k) \equiv \mathscr{SK}(M_i)$ for all $i \geqslant k$.

Now as there are only a finite number of closures in $\mathscr{SK}(M_k)$, and since the reductions within these closures are independant in the sense that they occur in parallel positions, by König's Lemma, an infinite derivation must take place within the same closure in $\mathscr{SK}(M_k)$. Thus there must exist a position p in $\mathscr{SK}(M_k)$ and a context C such that $M_k \equiv C\{P\langle y := Q_k\rangle\}_p$ and

$$\begin{array}{lll} M_1 \to_{\lambda\mathtt{x}} M_2 \to_{\lambda\mathtt{x}} \ldots \to_{\lambda\mathtt{x}} M_k & \equiv & C\{P\langle y := Q_k\rangle\}_p \\ & \overset{v}{\to}_{\lambda\mathtt{x}} & C\{P\langle y := Q_{k+1}\rangle\}_p \\ & \overset{v}{\to}_{\lambda\mathtt{x}} & C\{P\langle y := Q_{k+2}\rangle\}_p \\ & & . \\ & & . \end{array}$$

is an infinite $\lambda\mathtt{x}$-reduction sequence starting from M. □

Definition 3.3 (Derivation ordering) Let $\mathscr{D}$ and $\mathscr{E}$ be two infinite derivations starting from a term M_1. Then $\mathscr{D} : M_1 \to_{\lambda\mathtt{x},p_1} M_2 \to_{\lambda\mathtt{x},p_2} M_3 \ldots M_n \to_{\lambda\mathtt{x},p_n} M_{n+1} \ldots$ is said to be *smaller* than $\mathscr{E} : M_1 \to_{\lambda\mathtt{x},q_1} M_2 \to_{\lambda\mathtt{x},q_2} M_3 \ldots M_n \to_{\lambda\mathtt{x},q_n} M'_{n+1} \ldots$ if $p_i = q_i$ for $i < n$ and q_n is a proper prefix of p_n.

Before proceeding to the main proposition of this section we would like to discuss the relation with the work by Benaissa *et al.* for proving PSN (Benaissa *et al.* 1996).

— *What Benaissa et al. do for PSN.* For PSN they consider a minimal infinite $\lambda\mathtt{x}$-reduction sequence starting from a *pure* term. Thus they can *always* trace back the

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

closure (guaranteed to exist by Lemma 3.6) to its (unique) point of creation (here the *Beta*-rule) and obtain a *smaller* derivation than the original one, hence contradicting minimality.

— *What we do for perpetuality*. Suppose $M \to_{\mathtt{x}} N$ and $N \in SN_{\lambda\mathtt{x}}$. Given a minimal infinite λx-reduction sequence starting from M, we trace back the closure (guaranteed to exist by Lemma 3.6). But now there are *two possible* situations as dictated by Lemma 3.4
  - either, the closure was created by a *Beta*-rule. We then argue as above and contradict minimality.
  - or, (an ancestor of) the body of the closure, say Q, *belongs to the original term* M. But then, since $M \to_{\mathtt{x}} N$, we may reason as follows:
    - either, the substitution body Q *also occurs* in N, in which case, since $N \in SN_{\lambda\mathtt{x}}$, we arrive at a contradiction,
    - or, Q is a subterm of garbage, that is, $M \to_{gc} N$ and $Q \subseteq P$ where P is the substitution body eliminated by gc, then by requiring that all eliminated garbage be λx- strongly normalizing terms, we arrive at a contradiction,
    - or, the reduction step $M \to_{\mathtt{x}} N$ takes place inside the substitution body Q, that is, $Q \to_{\mathtt{x}} Q'$ for some $Q' \subset N$. Then we may apply inductively the same reasoning in order to obtain a contradiction.

These ideas may be formalized as follows.

Remark 3.7 Suppose $M \equiv C\{P\langle y := Q\rangle\} \to_{\mathtt{x}} N$. Then there are two possibilities:

1 either, the reduction step $M \equiv C\{P\langle y := Q\rangle\} \to_{\mathtt{x}} N$ takes place in Q, that is, $N \equiv C\{P\langle y := Q'\rangle\}$ and $Q \to_{\mathtt{x}} Q'$,
2 or, it does not take place inside Q.

In the second case we then have that the substitution body Q also occurs in N, or it was erased as a result of applying the gc-rule (Q is a subterm of garbage erased by gc). This observation will be made more precise in the following lemma.

Lemma 3.8 Let $M \equiv C\{P\langle y := Q\rangle\}_q \to_{\mathtt{x},p} N$ with $q.2 \not< p$ (that is, the x-reduction step does not take place inside Q) and let S be the term (substitution body) eliminated by the gc-rule if it was applied. Then either Q occurs in N or $Q \subseteq S$.

Proof. The proof is by induction on the context C. □

Lemma 3.9 Let $M, N \in \mathscr{T}_{\lambda\mathtt{x}}$.

1 Suppose $M \to_{\mathtt{x}\setminus gc} N$ and $N \in SN_{\lambda\mathtt{x}}$. Then $M \in SN_{\lambda\mathtt{x}}$.
2 Suppose $M \equiv D\{P_1\langle z := P_2\rangle\}_q \to_{gc,q} D\{P_1\}_q \equiv N$ and $P_2, N \in SN_{\lambda\mathtt{x}}$. Then $M \in SN_{\lambda\mathtt{x}}$.

Proof. Both items are proved simultaneously by (course of values) induction on the number of occurrences n of the closure operator in M. Note that since $M \to_{\mathtt{x}} N$ by hypothesis, there must be at least one occurrence of the closure operator in M.

— $n = 1$. Suppose that $M \notin SN_{\lambda\mathtt{x}}$ and let $\mathscr{D} : M \equiv M_1 \to_{\lambda\mathtt{x}} M_2 \to_{\lambda\mathtt{x}} M_3 \ldots$ be a *minimal infinite derivation* starting from M. Now, since $N \in SN_{\lambda\mathtt{x}}$, we have $\mathtt{x}(N) \in SN_{\beta}$, and

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

hence $\mathrm{x}(M) \in SN_\beta$ (since $\mathrm{x}(M) \equiv \mathrm{x}(N)$). Then, by Lemma 3.6, we may construct the following infinite $\lambda\mathrm{x}$-reduction sequence,

$$\begin{array}{lll} \mathscr{D}' : M_1 \to_{\lambda\mathrm{x}} M_2 \to_{\lambda\mathrm{x}} \ldots \to_{\lambda\mathrm{x}} M_k & \equiv & C\{P\langle y := Q_k\rangle\}_p \\ & \stackrel{v}{\to}_{\lambda\mathrm{x}} & C\{P\langle y := Q_{k+1}\rangle\}_p \\ & \stackrel{v}{\to}_{\lambda\mathrm{x}} & C\{P\langle y := Q_{k+2}\rangle\}_p \\ & & \vdots \end{array}$$

for some $k \in \mathbb{N}$, position p, variable y, context C and $\lambda\mathrm{x}$-terms $P, Q_k, Q_{k+1}, Q_{k+2}, \ldots$. Note that the sequence $(Q_k, Q_{k+1}, Q_{k+2}, \ldots)$ is an infinite $\lambda\mathrm{x}$-reduction sequence.
Now, by the closure tracing Lemma 3.4, there are two possibilities for the origin of the closure $\langle y := Q_k\rangle$:

– either, it was created some time earlier by an application of the *Beta*-rule, that is, there exists $j < k$, a context C' and a position p_j such that

$$M_j \equiv C'\{(\lambda y.P')Q\}_{p_j} \to_{Beta,p_j} C'\{P'\langle y := Q\rangle\}_{p_j} \equiv M_{j+1}$$

with $Q \stackrel{*}{\to}_{\lambda\mathrm{x}} Q_k$. But then we may construct the infinite $\lambda\mathrm{x}$-reduction sequence

$$\begin{array}{lll} \mathscr{D}'' : M_1 \to_{\lambda\mathrm{x}} M_2 \ldots \to_{\lambda\mathrm{x}} M_j & \equiv & C'\{(\lambda y.P')Q\}_{p_j} \\ & \stackrel{*}{\to}_{\lambda\mathrm{x}} & C'\{(\lambda y.P')Q_k\}_{p_j} \\ & \stackrel{v}{\to}_{\lambda\mathrm{x}} & C'\{(\lambda y.P')Q_{k+1}\}_{p_j} \\ & \stackrel{v}{\to}_{\lambda\mathrm{x}} & C'\{(\lambda y.P')Q_{k+2}\}_{p_j} \\ & & \vdots \end{array}$$

Note that $\mathscr{D}''$ is smaller than $\mathscr{D}$ since at step j we reduce a proper subterm of $(\lambda y.P')Q$ and obtain, nonetheless, an infinite $\lambda\mathrm{x}$-reduction sequence. Thus we contradict the minimality of $\mathscr{D}$.

– or, (an ancestor of) the body of the closure, *belongs to the original term* M, that is, $M \equiv C'\{P'\langle y := Q\rangle\}$ for some context C' and terms P' and Q with $Q \stackrel{*}{\to}_{\lambda\mathrm{x}} Q_k$. Here we have two further cases to consider:

  1 either the reduction step $M \equiv C'\{P'\langle y := Q\rangle\} \to_{\mathrm{x}} N$ takes place in Q, that is, $N \equiv C'\{P'\langle y := Q'\rangle\}$ and $Q \to_{\mathrm{x}} Q'$,

  2 or it does not take place inside Q.

  But the first case is not possible since $n = 1$, and therefore there are no occurrences of the closure operator in Q. So the reduction step $M \equiv C'\{P'\langle y := Q\rangle\} \to_{\mathrm{x}} N$ does not take place inside Q, and therefore by Lemma 3.8, we have that either Q occurs in N (which is in $SN_{\lambda\mathrm{x}}$) or is a subterm of garbage (that is, $Q \subseteq P_2$), and hence also is in $SN_{\lambda\mathrm{x}}$ as the additional requirement on application of gc-rule states. Thus we arrive at a contradiction (closure tracing has determined that there is an infinite $\lambda\mathrm{x}$-reduction sequence starting from Q and, on the other hand, we have $Q \in SN_{\lambda\mathrm{x}}$).

— $n > 1$. We start off as in the previous case; we shall repeat the argument for the sake of clarity. Suppose that $M \notin SN_{\lambda\mathrm{x}}$ and let $\mathscr{D} : M \equiv M_1 \to_{\lambda\mathrm{x}} M_2 \to_{\lambda\mathrm{x}} M_3 \ldots$ be a *minimal*

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

infinite derivation starting from M. Now, since $N \in SN_{\lambda x}$, we have $x(N) \in SN_{\beta}$, and hence $x(M) \in SN_{\beta}$ (since $x(M) \equiv x(N)$). Then by Lemma 3.6 we may construct the following infinite λx-reduction sequence,

$$\begin{array}{lll} \mathscr{D}' : M_1 \to_{\lambda x} M_2 \to_{\lambda x} \ldots \to_{\lambda x} M_k & \equiv & C\{P\langle y := Q_k\rangle\}_p \\ & \overset{v}{\to}_{\lambda x} & C\{P\langle y := Q_{k+1}\rangle\}_p \\ & \overset{v}{\to}_{\lambda x} & C\{P\langle y := Q_{k+2}\rangle\}_p \\ & & . \\ & & . \end{array}$$

for some $k \in \mathbb{N}$, position p, variable y, context C and λx-terms $P, Q_k, Q_{k+1}, Q_{k+2}, \ldots$. Note that the sequence $(Q_k, Q_{k+1}, Q_{k+2}, \ldots)$ is an infinite λx-reduction sequence.
Now by the closure tracing Lemma 3.4 there are two possibilities for the origin of the closure $\langle y := Q_k\rangle$:

- either, it was created some time earlier by an application of the *Beta*-rule, in which case we may proceed as above in order to arrive at a contradiction,
- or, (an ancestor of) the body of the closure, *belongs to the original term* M, that is, $M \equiv C'\{P'\langle y := Q\rangle\}$ for some context C' and terms P' and Q with $Q \overset{*}{\to}_{\lambda x} Q_k$. Here we have two further cases to consider:
  1 either the reduction step $M \equiv C'\{P'\langle y := Q\rangle\} \to_x N$ takes place in Q, that is, $N \equiv C'\{P'\langle y := Q'\rangle\}$ and $Q \to_x Q'$,
  2 or it does not take place inside Q.

  By Lemma 3.8, in the second case, we have that either Q also occurs in N (which is in $SN_{\lambda x}$) or is a subterm of garbage (that is, $Q \subseteq P_2$) and hence also is in $SN_{\lambda x}$, as the additional requirement on application of *gc*-rule states. Thus we arrive at a contradiction (closure tracing has determined that there is an infinite λx-reduction sequence starting from Q and, on the other hand, we have $Q \in SN_{\lambda x}$).
  We are left to consider the first case, that is, the case where $N \equiv C'\{P'\langle y := Q'\rangle\}$ and $Q \to_x Q'$. But since $N \in SN_{\lambda x}$, we have $Q' \in SN_{\lambda x}$, and we may apply the induction hypothesis and obtain $Q \in SN_{\lambda x}$. Hence we obtain a contradiction. □

Proposition 3.10 (Perpetuality proposition) Let $M, N \in \mathscr{T}_{\lambda x}$. Then:

1 If $M \equiv P_1\langle x := P_2\rangle P_3 \ldots P_n \in SN_{\lambda x}$, then $N \equiv (\lambda x.P_1)P_2 \ldots P_n \in SN_{\lambda x}$.
2 If $M \to_{x \setminus gc} N$ and $N \in SN_{\lambda x}$, then $M \in SN_{\lambda x}$.
3 If $M \equiv D\{P_1\langle z := P_2\rangle\}_q \to_{gc,q} D\{P_1\}_q \equiv N$ and $P_2, N \in SN_{\lambda x}$, then $M \in SN_{\lambda x}$.

Proof. The last two items are proved in the same way as in Lemma 3.9. For the first item we consider the term $P_1\langle x := P_2\rangle P_3 \ldots P_n \in SN_{\lambda x}$. Then $P_1, \ldots, P_n \in SN_{\lambda x}$. Thus any infinite derivation starting from the term $(\lambda x.P_1)P_2 \ldots P_n$ must have the form,

$$\begin{array}{lll} (\lambda x.P_1)P_2 \ldots P_n & \overset{*}{\to}_{\lambda x} & (\lambda x.P_1')P_2' \ldots P_n' \\ & \to_{Beta} & P_1'\langle x := P_2'\rangle P_3' \ldots P_n' \\ & \to_{\lambda x} & \ldots \end{array}$$

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

where $P_i \overset{*}{\rightarrow}_{\lambda x} P_i'$ for $i \in 1..n$. Then there is an infinite reduction sequence,

$$\begin{array}{lll} P_1\langle x := P_2\rangle P_3 \dots P_n & \overset{*}{\rightarrow}_{\lambda x} & P_1'\langle x := P_2'\rangle P_3' \dots P_n' \\ & \rightarrow_{\lambda x} & \dots \end{array}$$

contradicting the hypothesis. □

Remark 3.11 Note that the above technique may be applied to the explicit substitution formulation of the substitution lemma. More precisely, we have that if $y \notin FV(P)$ and $C\{M\langle x := P\rangle\langle y := N\langle x := P\rangle\rangle\}_p$ is λx-strongly normalizing, then $C\{M\langle y := N\rangle\langle x := P\rangle\}_p$ is also λx-strongly normalizing. Obviously, this does not mean that we may add the inferred rule (that is, $M\langle y := N\rangle\langle x := P\rangle \rightarrow_{co} M\langle x := P\rangle\langle y := N\langle x := P\rangle\rangle$) to λx without (trivially) losing strong normalization.

4. Some applications of the perpetuality analysis

This section considers some applications of the perpetuality analysis elaborated in Section 3: an inductive characterization of $SN_{\lambda x}$, two perpetual reduction strategies for λx, and strong normalization by reducibility for a polymorphic lambda calculus with explicit substitutions F_{es} (Bonelli 1999). This last issue will be considered in detail in Section 5. The technique presented in this section is based on similar results obtained for the λ-calculus with β-reduction (van Raamsdonk and Severi 1995).

4.1. *An inductive characterization of* $SN_{\lambda x}$

We give an inductive characterization of the terms in $SN_{\lambda x}$ by combining the technique presented in van Raamsdonk and Severi (1995) and the Perpetuality Proposition (Proposition 3.10). This characterization will be used in the proof of perpetuality of some reduction strategies studied in Section 4.2.

Lemma 4.1 Every $M \in \mathscr{T}_{\lambda x}$ is of one of the following forms:

1 $xP_1 \dots P_n$
2 $\lambda x.P$
3 $(\lambda x.P)P_1 \dots P_n$
4 $P_0\langle x_1 := Q_1\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n$ where $m \geqslant 1$ and P_0 is not a closure (that is, it is either a variable, an abstraction or an application).

Proof. The proof is by induction on the term M. □

Definition 4.1 Let $\mathscr{SN} \subseteq \mathscr{T}_{\lambda x}$ be the smallest set closed under the clauses

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

$$\frac{P_1,\ldots,P_n \in \mathcal{SN} \quad x \in \mathcal{X}}{xP_1\ldots P_n \in \mathcal{SN}}\,(R1) \qquad \frac{P \in \mathcal{SN}}{\lambda x.P \in \mathcal{SN}}\,(R2)$$

$$\frac{M\langle x := N\rangle P_1\ldots P_n \in \mathcal{SN}}{(\lambda x.M)\ NP_1\ldots P_n \in \mathcal{SN}}\,(R3) \qquad \frac{N^\star P_1\ldots P_n \in \mathcal{SN} \quad x \in \mathcal{X}}{x\langle x := N\rangle^\star P_1\ldots P_n \in \mathcal{SN}}\,(R4)$$

$$\frac{y^\star P_1\ldots P_n \in \mathcal{SN} \quad N \in \mathcal{SN} \quad x \in \mathcal{X}}{y\langle x := N\rangle^\star P_1\ldots P_n \in \mathcal{SN}}\,(R5) \qquad \frac{(\lambda y.M\langle x := N\rangle)^\star P_1\ldots P_n \in \mathcal{SN}}{(\lambda y.M)\langle x := N\rangle^\star P_1\ldots P_n \in \mathcal{SN}}\,(R7)$$

$$\frac{(M_1\langle x := N\rangle M_2\langle x := N\rangle)^\star P_1\ldots P_n \in \mathcal{SN}}{(M_1M_2)\langle x := N\rangle^\star P_1\ldots P_n \in \mathcal{SN}}\,(R6)$$

where clause ($R5$) is subject to the restriction $x \not\equiv y$.

Each clause is quantified over n and m (if m also occurs in it). Thus, for example, clause ($R5$) should be read as: for all $m \geqslant 0$ and $n \geqslant 0$, if $N \in \mathcal{SN}$ and $y\langle x_1 := N_1\rangle\ldots\langle x_m := N_m\rangle P_1\ldots P_n \in \mathcal{SN}$ and $y \not\equiv x$, then $y\langle x := N\rangle\langle x_1 := N_1\rangle\ldots\langle x_m := N_m\rangle P_1\ldots P_n \in \mathcal{SN}$.

Lemma 4.2 $SN_{\lambda x} = \mathcal{SN}$.

Proof.

— $SN_{\lambda x} \subseteq \mathcal{SN}$. Let $M \in SN_{\lambda x}$. We prove by induction on $(maxred(M), |M|)$, using the usual lexicographic ordering, that $M \in \mathcal{SN}$. By Lemma 4.1, we have the following cases to consider:

- $M \equiv xP_1\ldots P_n$. Then, since M is λx-strongly normalizing, $P_1,\ldots,P_n$ are λx-strongly normalizing. By the induction hypothesis, we have that $P_1,\ldots,P_n \in \mathcal{SN}$. Thus by clause ($R1$), we obtain $xP_1\ldots P_n \in \mathcal{SN}$ (note that this case includes the base case, that is, all variables are in $\mathcal{SN}$).
- $M \equiv \lambda x.P$. As above P is λx-strongly normalizing, and therefore, by the induction hypothesis, $P \in \mathcal{SN}$. Using clause ($R2$), we conclude that $\lambda x.P \in \mathcal{SN}$.
- $M \equiv (\lambda x.P_0)P_1\ldots P_n$. Then, since $M \rightarrow_{Beta} P_0\langle x := P_1\rangle P_2\ldots P_n$, we have that $P_0\langle x := P_1\rangle P_2..P_n \in SN_{\lambda x}$. By the induction hypothesis, $P_0\langle x := P_1\rangle P_2\ldots P_n \in \mathcal{SN}$, and thus by clause ($R3$), we conclude that $M \in \mathcal{SN}$.
- $M \equiv P_0\langle x_1 := Q_1\rangle\ldots\langle x_m := Q_m\rangle P_1\ldots P_n$ where $m \geqslant 1$ and P_0 is not a closure (that is, not of the form $P'\langle z := Q'\rangle$). Thus we have the following cases to consider:
  - $P_0 \equiv x_1$. Then $Q_1\langle x_2 := Q_2\rangle\ldots\langle x_m := Q_m\rangle P_1\ldots P_n \in SN_{\lambda x}$, and by the induction hypothesis this term belongs to $\mathcal{SN}$. Then, by clause ($R4$), we obtain that $M \in \mathcal{SN}$.
  - $P_0 \equiv y \not\equiv x_1$. Then $y\langle x_2 := Q_2\rangle\ldots\langle x_m := Q_m\rangle P_1\ldots P_n \in SN_{\lambda x}$, and by the induction hypothesis it is also an element of $\mathcal{SN}$. Also, since $M \in SN_{\lambda x}$, we have $Q_1 \in SN_{\lambda x}$, and by the induction hypothesis, $Q_1 \in \mathcal{SN}$. Then, by clause ($R5$), we obtain that $M \in \mathcal{SN}$.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

- $P_0 \equiv \lambda y.N$. Then $(\lambda y.N\langle x_1 := Q_1\rangle)\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in SN_{\lambda x}$. By the induction hypothesis, $(\lambda y.N\langle x_1 := Q_1\rangle)\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in \mathscr{SN}$. Then, using clause (*R*7), we obtain that $M \in \mathscr{SN}$.
- $P_0 \equiv N_1 N_2$. Then we have $(N_1\langle x_1 := Q_1\rangle N_2\langle x_1 := Q_1\rangle)\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in SN_{\lambda x}$. By the induction hypothesis, $(N_1\langle x_1 := Q_1\rangle N_2\langle x_1 := Q_1\rangle)\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in \mathscr{SN}$. Using clause (*R*6), we may conclude that $M \in \mathscr{SN}$.

— $\mathscr{SN} \subseteq SN_{\lambda x}$. We use induction on the derivation of $M \in \mathscr{SN}$.

- $M \equiv xP_1 \dots P_n$ where $P_1, \dots, P_n \in \mathscr{SN}$. By the induction hypothesis, we have $P_1, \dots, P_n \in SN_{\lambda x}$. And therefore $xP_1 \dots P_n$ is also λx-strongly normalizing.
- $M \equiv \lambda x.P$ where $P \in \mathscr{SN}$. This is similar to the previous case.
- $M \equiv (\lambda x.P_0)P_1 \dots P_n$ where $P_0\langle x := P_1\rangle P_2 \dots P_n \in \mathscr{SN}$. Then, by the induction hypothesis, we have $P_0\langle x := P_1\rangle P_2 \dots P_n \in SN_{\lambda x}$. Thus, by the Perpetuality Proposition 3.10(1), $M \in SN_{\lambda x}$.
- $M \equiv x_1\langle x_1 := Q_1\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n$ where $Q_1\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in \mathscr{SN}$. Then, by the induction hypothesis, $Q_1\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in SN_{\lambda x}$, and thus by the Perpetuality Proposition 3.10(2), we may conclude that $M \in SN_{\lambda x}$.
- $M \equiv y\langle x_1 := Q_1\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n$ where $y \not\equiv x_1$, $Q_1 \in \mathscr{SN}$ and $y\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in \mathscr{SN}$. Then, by the induction hypothesis, $Q_1 \in SN_{\lambda x}$ and $y\langle x_2 := Q_2\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n \in SN_{\lambda x}$. And, applying the Perpetuality Proposition 3.10(3), we obtain $M \in SN_{\lambda x}$.
- $M \equiv (N_1 N_2)\langle x_1 := Q_1\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n$. This is similar to the above cases using the Perpetuality Proposition 3.10(2).
- $M \equiv (\lambda y.N)\langle x_1 := Q_1\rangle \dots \langle x_m := Q_m\rangle P_1 \dots P_n$. This is similar to the above cases using the Perpetuality Proposition 3.10(2). □

Some variations of Definition 4.1 are possible. Rule (*R*5) may be replaced by the following rule (*R*5′) yielding the characterization $\mathscr{SN}^{-}$:

$$\frac{M^\star P_1 \dots P_n \in \mathscr{SN} \quad N \in \mathscr{SN} \quad x \in \mathscr{X}}{M\langle x := N\rangle^\star P_1 \dots P_n \in \mathscr{SN}} \ (R5')$$

with the restriction that $x \notin FV(M)$. Then Lemma 4.2 may be proved for $\mathscr{SN}^{-}$. As a consequence, we also obtain $\mathscr{SN} = \mathscr{SN}^{-}$.

Another possible characterization of $SN_{\lambda x}$, as reported in Xi (1996) for the λ-calculus with β-reduction, is $SN_{\lambda x} = \{M \in \mathscr{T}_{\lambda x} \mid \mathscr{H}(M) < \infty\}$ where

$$\mathscr{H}(M_1) = \max\{n \mid M_1 \triangleright M_2 \triangleright \dots \triangleright M_n\}$$

and $\triangleright$ is the union of the strict subterm relation and leftmost reduction for λx, that is, $\triangleright =_{def} \sqsupset \cup \rightarrow_l$ where $\lambda x.P \sqsupset P$, $PQ \sqsupset P$, $PQ \sqsupset Q$, $P\langle x := Q\rangle \sqsupset P$, $P\langle x := Q\rangle \sqsupset Q$. The proof relies on the Perpetuality Proposition.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

4.2. *Perpetual reduction strategies for* λx

Here we extend the perpetual reduction strategies presented in Bergstra and Klop (1982) and Barendregt *et al.* (1976) to the calculus of explicit substitutions λx, and, following van Raamsdonk and Severi (1995), use the characterization of $SN_{\lambda x}$ (Definition 4.1) to prove that they indeed constitute perpetual strategies. First, we recall some definitions taken from Barendregt (1984).

Definition 4.2

(i) A (one step) λx-*reduction strategy* is a function $\mathscr{F}(.) : \mathscr{T}_{\lambda x} \longrightarrow \mathscr{T}_{\lambda x}$ such that for all $M \in \mathscr{T}_{\lambda x}$ we have $M \rightarrow_{\lambda x} \mathscr{F}(M)$, unless M is in normal form, in which case $\mathscr{F}(M) \equiv M$.
(ii) A strategy is called *perpetual* if $\infty(M)$ implies $\infty(\mathscr{F}(M))$.

First we present the effective strategy $\mathscr{F}_\infty(.)$. Then the strategy $\mathscr{F}(.)$ will be considered.

Definition 4.3 (The strategy $\mathscr{F}_\infty(.)$) Let M be a term in $\mathscr{T}_{\lambda x}$ that is not in normal form. Let $M \equiv C\{\Delta\}$ where Δ is the leftmost λx^--redex of M. We define $\mathscr{F}_\infty(.)$ as follows:

$$
\begin{array}{lll}
\mathscr{F}_\infty(C\{(\lambda y.P)Q\}) & =_{def} & C\{P\langle y := Q\rangle\} \\
\mathscr{F}_\infty(C\{(\lambda y.P)\langle z := Q\rangle\}) & =_{def} & C\{\lambda y.P\langle z := Q\rangle\} \\
\mathscr{F}_\infty(C\{(P_1P_2)\langle y := Q\rangle\}) & =_{def} & C\{P_1\langle y := Q\rangle P_2\langle y := Q\rangle\} \\
\mathscr{F}_\infty(C\{y\langle y := Q\rangle\}) & =_{def} & C\{Q\} \\
\mathscr{F}_\infty(C\{z\langle y := Q\rangle\}) & =_{def} & \begin{cases} C\{z\} & \text{if } Q \text{ in normal form,} \\ C\{z\langle y := \mathscr{F}_\infty(Q)\rangle\} & \text{otherwise} \end{cases}
\end{array}
$$

In the last clause we have $y \not\equiv z$.

Proposition 4.3 $\mathscr{F}_\infty(.)$ is a perpetual reduction strategy for λx.

Proof. We prove that if $\mathscr{F}_\infty(M)$ is strongly normalizing, then M is strongly normalizing. For this, the characterization of Definition 4.1 will be used. Thus we assume $\mathscr{F}_\infty(M) \in \mathscr{SN}$ and show that $M \in \mathscr{SN}$ by induction on the number of derivation steps of $\mathscr{F}_\infty(M) \in \mathscr{SN}$. By Lemma 4.1, we have the following cases to consider:

— $M \equiv yP_1, \ldots P_n$. Then we have $\mathscr{F}_\infty(M) \equiv yP_1 \ldots P_{i-1}\mathscr{F}_\infty(P_i)P_{i+1} \ldots P_n$ where $P_1, \ldots, P_{i-1}$ are in normal form. Then, since $\mathscr{F}_\infty(M) \in \mathscr{SN}$, we have $\mathscr{F}_\infty(P_i) \in \mathscr{SN}$. Then, by the induction hypothesis, $P_i \in \mathscr{SN}$ and we can construct a derivation in $\mathscr{SN}$ of M.
— $M \equiv \lambda y.P$. Then we have $\mathscr{F}_\infty(M) \equiv \lambda y.\mathscr{F}_\infty(P)$. By the induction hypothesis, $P \in \mathscr{SN}$. Hence, $M \in \mathscr{SN}$.
— $M \equiv (\lambda y.P)P_1 \ldots P_n$. Then we have $\mathscr{F}_\infty(M) \equiv P\langle y := P_1\rangle P_2 \ldots P_n$. Since $\mathscr{F}_\infty(M) \in \mathscr{SN}$, using clause $(R3)$, we obtain that $M \in \mathscr{SN}$.
— $M \equiv P_0\langle y_1 := Q_1\rangle \ldots \langle Q_m := Q_m\rangle P_1 \ldots P_n$ with $m \geqslant 1$ and P_0 not a closure (that is, not of the form $P'\langle z := Q'\rangle$). We consider four further cases:
  – $P_0 \equiv y_1$. Then $\mathscr{F}_\infty(M) \equiv Q_1\langle y_2 := Q_2\rangle \ldots \langle y_m := Q_m\rangle P_1 \ldots P_n$. Then, by using clause $(R4)$, we obtain $M \in \mathscr{SN}$.
  – $P_0 \equiv z \not\equiv y_1$. Then we have two cases to consider:

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

- Q_1 is in normal form. Then $\mathscr{F}_\infty(M) \equiv z\langle y_2 := Q_2\rangle \ldots \langle y_m := Q_m\rangle P_1 \ldots P_n$. Since $Q_1 \in SN_{\lambda \mathtt{x}}$, we have $Q_1 \in \mathscr{SN}$. And then by using clause ($R5$), we obtain $M \in \mathscr{SN}$.
- Q_1 is not in normal form. Then we have $\mathscr{F}_\infty(M) \equiv z\langle y_1 := \mathscr{F}_\infty(Q_1)\rangle \ldots \langle y_m := Q_m\rangle P_1 \ldots P_n$. Now, since $\mathscr{F}_\infty(M) \in \mathscr{SN}$, we have $\mathscr{F}_\infty(Q_1) \in \mathscr{SN}$, and hence, by the induction hypothesis, $Q_1 \in \mathscr{SN}$. Then we can construct a derivation in $\mathscr{SN}$ of M.

– $P_0 \equiv P_0' P_0''$. Then $\mathscr{F}_\infty(M) \equiv (P_0'\langle y_1 := Q_1\rangle P_0''\langle y_1 := Q_1\rangle)\langle y_2 := Q_2\rangle \ldots \langle y_m := Q_m\rangle P_1 \ldots P_n$. Then, by using clause ($R6$), we obtain $M \in \mathscr{SN}$.

– $P_0 \equiv \lambda z.P_0'$. Then $\mathscr{F}_\infty(M) \equiv (\lambda z.P_0'\langle y_1 := Q_1\rangle)\langle y_2 := Q_2\rangle \ldots \langle y_m := Q_m\rangle P_1 \ldots P_n$. Then, by using clause ($R7$), we obtain $M \in \mathscr{SN}$. □

Remark 4.4 Let F be a λx-reduction strategy and define

$$L_F(M) =_{def} \min\{n \mid F^n(M) \text{ is in } \lambda\text{x-normal form}\}$$

where $F^0(M) =_{def} M$ and $F^{n+1}(M) =_{def} F(F^n(M))$. Then F is said to be *maximal* iff $L_F(M) = maxred(M)$. Is $\mathscr{F}_\infty(.)$ maximal? The perpetual reduction strategy F_∞ for the λ-calculus (Barendregt *et al.* 1976) has been shown to be a maximal strategy (Sørensen 1996).

Definition 4.4 (The strategy $\mathscr{F}(.)$) Let M be a term in $\mathscr{T}_{\lambda \mathtt{x}}$ that is not in normal form. Let $M \equiv C\{\Delta\}$ where Δ is the leftmost $\lambda\mathtt{x}^-$-redex of M. We define $\mathscr{F}(.)$ as follows:

$$\begin{array}{lll}
\mathscr{F}(C\{(\lambda y.P)Q\}) & =_{def} & C\{P\langle y := Q\rangle\} \\
\mathscr{F}(C\{(\lambda z.P)\langle y := Q\rangle\}) & =_{def} & C\{\lambda z.P\langle y := Q\rangle\} \\
\mathscr{F}(C\{(P_1P_2)\langle y := Q\rangle\}) & =_{def} & C\{P_1\langle y := Q\rangle P_2\langle y := Q\rangle\} \\
\mathscr{F}(C\{y\langle y := Q\rangle\}) & =_{def} & C\{Q\} \\
\mathscr{F}(C\{z\langle y := Q\rangle\}) & =_{def} & \begin{cases} C\{z\} & \text{if } Q \in SN_{\lambda \mathtt{x}} \\ C\{z\langle y := \mathscr{F}(Q)\rangle\} & \text{if } Q \notin SN_{\lambda \mathtt{x}} \end{cases}
\end{array}$$

In the last clause we have $y \not\equiv z$.

Proposition 4.5 $\mathscr{F}(.)$ is a perpetual reduction strategy for λx.

Proof. The proof is similar to that of Proposition 4.3. □

Note that these strategies are not refinings of the original ones (those defined on the λ-calculus with β-reduction) in the following sense: if we start from a pure term M, the strategy for λx yields a rewrite sequence consisting of a *Beta*-reduction step followed by x-reduction to x-normal form, which corresponds to the β-reduction step induced by the original strategy. Before illustrating with an example we recall the strategy $F_\infty(.)$ (Barendregt *et al.* 1976).

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Definition 4.5 ($F_\infty(.)$) Let M be a term in $\mathscr{T}_\lambda$ that is not in β-normal form (denoted $M \notin NF_\beta$). Let $M \equiv C\{(\lambda y.P)Q\}$ where $(\lambda y.P)Q$ is the leftmost β-redex of M. Then $F_\infty(M)$ is defined as follows:

$$F_\infty(C\{(\lambda y.P)Q\}) \quad =_{def} \quad \begin{cases} C\{P\{y \leftarrow Q\}\} & \text{if } y \in FV(P) \text{ or } Q \in NF_\beta \\ C\{(\lambda y.P)F_\infty(Q)\} & \text{if } y \notin FV(P) \text{ and } Q \notin NF_\beta. \end{cases}$$

Let $M \in \mathscr{T}_\lambda$. More generally, one may wonder whether there exists a positive integer n such $M \xrightarrow{+}_{\lambda\mathtt{x}} \mathscr{F}^n_\infty(M)$ and $\mathscr{F}^n_\infty(M) \equiv F_\infty(M)$. Consider the term $M \equiv (\lambda x.\omega\omega x)z$ where $\omega \equiv \lambda y.yy$. Then $F_\infty(M) \equiv \omega\omega z$. On the other hand, for no positive integer n is $\mathscr{F}^n_\infty(M)$ a pure term (and hence $\mathscr{F}^n_\infty(M) \not\equiv F_\infty(M)$ for all positive integers n) as the following portion of reduction of the strategy $\mathscr{F}_\infty(.)$ suggests:

$$\begin{array}{ll} & M \\ \rightarrow_{Beta} & (\omega\omega x)\langle x := z\rangle \\ \rightarrow_{app} & (\omega\omega)\langle x := z\rangle x\langle x := z\rangle \\ \rightarrow_{app} & \omega\langle x := z\rangle\omega\langle x := z\rangle x\langle x := z\rangle \\ \rightarrow_{lam} & (\lambda y.(yy)\langle x := z\rangle)\omega\langle x := z\rangle x\langle x := z\rangle \\ \rightarrow_{Beta} & (yy)\langle x := z\rangle\langle y := \omega\langle x := z\rangle\rangle x\langle x := z\rangle \\ \rightarrow_{app} & (y\langle x := z\rangle y\langle x := z\rangle)\langle y := \omega\langle x := z\rangle\rangle x\langle x := z\rangle \\ \rightarrow_{app} & y\langle x := z\rangle\langle y := \omega\langle x := z\rangle\rangle y\langle x := z\rangle\langle y := \omega\langle x := z\rangle\rangle x\langle x := z\rangle \\ \rightarrow_{gc} & y\langle y := \omega\langle x := z\rangle\rangle y\langle x := z\rangle\langle y := \omega\langle x := z\rangle\rangle x\langle x := z\rangle \\ \rightarrow_{var} & \omega\langle x := z\rangle y\langle x := z\rangle\langle y := \omega\langle x := z\rangle\rangle x\langle x := z\rangle \\ & \ldots \end{array}$$

5. The F-calculus with explicit substitutions

As a final application of our studies in perpetuality for $\lambda\mathtt{x}$ we will formulate a polymorphic lambda calculus with explicit substitutions called F_{es} and prove the properties of subject reduction and strong normalization (of its typed version). The presentation is based on Bonelli (1999) with the improvement of incorporating a 'pure' reducibility proof of strong normalization in the sense that the characterization (Bloo and Geuvers 1998)

$$SN_{\lambda\mathtt{x}} = \{M \in \mathscr{T}_{\lambda\mathtt{x}} | \forall\ N \subseteq M,\ \mathtt{x}(N) \in SN_\beta\}$$

used in Bonelli (1999) in order to show strong normalization of F_{es} is no longer needed. The first subsection presents the untyped F-calculus with explicit substitutions and then introduces its typed version. Then the subject reduction property followed by strong normalization is considered.

5.1. *The untyped F-calculus with explicit substitutions*

In this subsection we introduce the polymorphic lambda calculus (Girard *et al.* 1989) with explicit substitutions. This reduction system, which we shall call F_{es}, will first be introduced as an untyped calculus in the sense that no typing rules for terms will be given; Section 5.2 will deal with its typed version.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Let $\mathscr{V}$ be an infinite set of type variables $s, t, u, \ldots$, and $\mathscr{X}$ be an infinite set of term variables (referred to simply as variables) $x, y, z, \ldots$.

Definition 5.1 The set of *types* and *terms* (referred to, without distinction, as raw terms) of the F_{es}-calculus is defined by the following two grammars:

$$\begin{array}{ll} \textit{types} & \sigma ::= t \mid \sigma_1 \supset \sigma_2 \mid \forall t.\sigma \mid \sigma[t := \sigma] \\ \textit{terms} & M ::= x \mid \lambda x : \sigma.M \mid \Lambda t.M \mid MN \mid M\sigma \mid M\langle x := N\rangle \mid M[t := \sigma] \end{array}$$

The operator $.\langle . := .\rangle$ is called the *term substitution operator* and $.[. := .]$ the *type substitution operator*. A term that does not contain occurrences of a (term or type) substitution operator is called a *pure term*. Likewise, a type that does not contain occurrences of the type substitution operator is called a *pure type*.

We use $\mathscr{RT}$ to denote the set of all raw terms, that is, those generated by the *terms* and *types* grammars, and $\mathscr{T}$ the set of terms in $\mathscr{RT}$, that is, those generated by the *terms* grammar.

Letters $\rho, \sigma, \tau, \ldots$ are used for types, letters $M, N, O, P, Q, \ldots$ are used for terms and letters $A, B, C, \ldots$ are used for raw terms.

The set of free variables of a term M is denoted $FV(M)$ and the set of free type variables of a raw term A is denoted $FTV(A)$. We also use $BV(M)$ and $BTV(A)$ to denote the set of bound variables of M and bound type variables of A, respectively. Also, we sometimes use $BTV(A, B)$ as a shorthand for $BTV(A) \cup BTV(B)$. These notions are defined as usual; we give below the definition of FTV as an example.

Definition 5.2 Let A be a raw term. The set of *free type variables of A* is defined as follows:

$$\begin{array}{lll} FTV(t) & =_{def} & \{t\} \\ FTV(\sigma \supset \tau) & =_{def} & FTV(\sigma) \cup FTV(\tau) \\ FTV(\forall t.\sigma) & =_{def} & FTV(\sigma) \setminus \{t\} \\ FTV(\sigma[t := \tau]) & =_{def} & (FTV(\sigma) \setminus \{t\}) \cup FTV(\tau) \\ FTV(x) & =_{def} & \varnothing \\ FTV(\lambda x : \sigma.P) & =_{def} & FTV(\sigma) \cup FTV(P) \\ FTV(\Lambda t.P) & =_{def} & FTV(P) \setminus \{t\} \\ FTV(PQ) & =_{def} & FTV(P) \cup FTV(Q) \\ FTV(P\sigma) & =_{def} & FTV(P) \cup FTV(\sigma) \\ FTV(P\langle x := Q\rangle) & =_{def} & FTV(P) \cup FTV(Q) \\ FTV(P[t := \sigma]) & =_{def} & (FTV(P) \setminus \{t\}) \cup FTV(\sigma) \end{array}$$

When dealing with calculi of explicit substitution, and in order to obtain first-order rewriting systems for the λ-calculus, it is not uncommon to use de Bruijn indices notation (see, for example, Abadi *et al.* (1991), Lescanne (1994), Ferreira *et al.* (1996) and Kamareddine and Ríos (1995)). Since our main objective is to concentrate on the properties of subject reduction and strong normalization, we have chosen to use the variable name based λx in order to minimize the 'noise' introduced by updating (that is, index updating in de Bruijn indices calculi) and thus provide the reader with a more intuitive setting.

We recall the familiar notions of substitution and α-conversion for terms (types are

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

treated similarly).

$$
\begin{array}{lll}
x\{x \leftarrow N\} & =_{def} & N \\
y\{x \leftarrow N\} & =_{def} & y, \text{ if } x \not\equiv y \\
(M_1 M_2)\{x \leftarrow N\} & =_{def} & M_1\{x \leftarrow N\} M_2\{x \leftarrow N\} \\
(\lambda x : \sigma.M)\{x \leftarrow N\} & =_{def} & \lambda x : \sigma.M \\
(\lambda y : \sigma.M)\{x \leftarrow N\} & =_{def} & \lambda y' : \sigma.M\{y \leftarrow y'\}\{x \leftarrow N\}, \\
 & & \text{if } y' \notin FV(N) \cup \{x\} \cup (FV(M) \setminus \{y\}) \\
(M_1\langle x := M_2\rangle)\{x \leftarrow N\} & =_{def} & M_1\langle x := M_2\{x \leftarrow N\}\rangle \\
(M_1\langle y := M_2\rangle)\{x \leftarrow N\} & =_{def} & M_1\{y \leftarrow y'\}\{x \leftarrow N\}\langle y' := M_2\{x \leftarrow N\}\rangle, \\
 & & \text{if } y' \notin FV(N) \cup \{x\} \cup (FV(M_1) \setminus \{y\}) \\
(\Lambda t.M)\{x \leftarrow N\} & =_{def} & \Lambda t.M\{x \leftarrow N\} \\
(M\sigma)\{x \leftarrow N\} & =_{def} & M\{x \leftarrow N\}\sigma \\
\tau[t := \sigma]\{x \leftarrow N\} & =_{def} & \tau[t := \sigma] \\
M[t := \sigma]\{x \leftarrow N\} & =_{def} & M\{t \leftarrow s\}\{x \leftarrow N\}[s := \sigma] \\
 & & \text{if } s \notin FTV(N) \cup (FTV(M) \setminus \{t\})
\end{array}
$$

Syntactical equality modulo α-conversion is thus the smallest equivalence relation verifying

$$
\begin{array}{lll}
\text{if } M \equiv P \text{ and } A \equiv B & \text{then} & MA \equiv PB \\
\text{if } M \equiv N, \sigma \equiv \tau \text{ and } y \notin FV(N) \setminus \{x\} & \text{then} & \lambda x : \sigma.M \equiv \lambda y : \tau.N\{x \leftarrow y\} \\
\text{if } M \equiv P, N \equiv Q \text{ and } y \notin FV(N) \setminus \{x\} & \text{then} & M\langle x := N\rangle \equiv P\{x \leftarrow y\}\langle y := Q\rangle \\
\text{if } M \equiv P \text{ and } s \notin FTV(P) \setminus \{t\} & \text{then} & \Lambda t.M \equiv \Lambda s.P\{t \leftarrow s\} \\
\text{if } A \equiv B, \sigma \equiv \tau \text{ and } s \notin FTV(B) \setminus \{t\} & \text{then} & A[t := \sigma] \equiv B\{t \leftarrow s\}[s := \tau]
\end{array}
$$

As in the untyped lambda calculus, we shall adopt the following variable convention: we assume that the names of bound (term or type) variables will always be chosen so that they differ from the free ones. Moreover, each occurrence of a (term or type) abstraction operator has a different binding variable.

Definition 5.3 (F_{es} and subsystems) The F_{es}-calculus is given by the rules in Figure 1. The F_{es}-calculus without the rules *Beta*2 and τ*Beta* is referred to as the ES-reduction system (or ES-calculus). The F_{es}-calculus without the *Beta*2 rule is called the $ES^\bullet$ reduction system. The third group of rules is called the ZT reduction system.

As regards the choice of the rules, perhaps it is worth mentioning that the *gc*-rule may be replaced by the more restricted garbage collection rule *rgc* without affecting the results. The same applies to the *ztgc*-rule and what might be called the restricted garbage collection rule for types $t[s := \tau] \rightarrow_{rztgc} t$. Also, the *ztvar*2-rule may be replaced by the more general rule $M[t := \tau] \rightarrow_{ztgc2} M$.

We shall now prove that the $ES^\bullet$ reduction system is strongly normalizing. Intuitively, one notes that the substitution calculus ES 'pushes' substitution operators deeper and deeper into a term/type until they are performed or eliminated. As for the rule τ*Beta* one notes that it eliminates an occurrence of the binder Λ. Thus we shall first prove that the ES reduction system is SN by interpreting it into a simpler calculus equipped with a well-founded reduction notion and showing that reduction is preserved. Then for the full system $ES^\bullet$ we use this result plus Lemmas A.1 (see the Appendix) and 5.4.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

$(\lambda x : \sigma.M)N$	$\rightarrow_{Beta2}$	$M\langle x := N\rangle$	
$(\Lambda t.M)\tau$	$\rightarrow_{\tau Beta}$	$M[t := \tau]$	
$(MN)\langle x := P\rangle$	$\rightarrow_{zapp}$	$M\langle x := P\rangle N\langle x := P\rangle$	
$(\lambda y : \sigma.M)\langle x := P\rangle$	$\rightarrow_{zlam}$	$\lambda y : \sigma.M\langle x := P\rangle$	
$x\langle x := P\rangle$	$\rightarrow_{zvar}$	P	
$M\langle x := P\rangle$	$\rightarrow_{zgc}$	M	$x \notin FV(M)$
$(M\sigma)\langle x := P\rangle$	$\rightarrow_{zappt}$	$M\langle x := P\rangle\sigma$	
$(\Lambda t.M)\langle x := P\rangle$	$\rightarrow_{zlamt}$	$\Lambda t.M\langle x := P\rangle$	
$(MN)[t := \tau]$	$\rightarrow_{ztapp}$	$M[t := \tau]N[t := \tau]$	
$(\lambda y : \sigma.M)[t := \tau]$	$\rightarrow_{ztlam}$	$\lambda y : \sigma[t := \tau].M[t := \tau]$	
$t[t := \tau]$	$\rightarrow_{ztvar1}$	τ	
$\sigma[t := \tau]$	$\rightarrow_{ztgc}$	σ	$t \notin FTV(\sigma)$
$s[t := \tau]$	$\rightarrow_{ztvar2}$	s	
$(M\sigma)[t := \tau]$	$\rightarrow_{ztappt}$	$M[t := \tau]\sigma[t := \tau]$	
$(\Lambda u.M)[t := \tau]$	$\rightarrow_{ztlamt}$	$\Lambda u.M[t := \tau]$	
$(\sigma_1 \supset \sigma_2)[t := \tau]$	$\rightarrow_{zt}$	$\sigma_1[t := \tau] \supset \sigma_2[t := \tau]$	
$(\forall u.\sigma)[t := \tau]$	$\rightarrow_{zt}$	$\forall u.\sigma[t := \tau]$	

Fig. 1. The F_{es}-calculus

Definition 5.4 The set of terms constructed from the alphabet $\mathscr{A} = \{\star, \lambda, \Lambda, \supset, ., [\![]\!]\}$, denoted $\mathscr{S}$, is defined by the grammar $a ::= \star \mid \lambda(a, a) \mid \Lambda a \mid a.a \mid a[\![a]\!] \mid a \supset a$. Also, we define the following well-founded ordering $[\![]\!] \gg . \gg \star, \lambda, \Lambda, \supset$ on the alphabet $\mathscr{A}$.

Assuming all symbols of $\mathscr{A}$ have multiset status, this well-founded ordering induces a well-founded Recursive Path Ordering (RPO) on the full set of terms $\mathscr{S}$ (Dershowitz 1982) (see the Appendix).

Definition 5.5 We define the translation $\mathscr{R}_1(.) : \mathscr{RT} \longrightarrow \mathscr{S}$ as

$\mathscr{R}_1(x) = \mathscr{R}_1(t)$	$=_{def}$	$\star$	$\mathscr{R}_1(M\langle x := N\rangle)$	$=_{def}$	$\mathscr{R}_1(M)[\![\mathscr{R}_1(N)]\!]$
$\mathscr{R}_1(MN)$	$=_{def}$	$\mathscr{R}_1(M).\mathscr{R}_1(N)$	$\mathscr{R}_1(M[t := \tau])$	$=_{def}$	$\mathscr{R}_1(M)[\![\mathscr{R}_1(\tau)]\!]$
$\mathscr{R}_1(M\sigma)$	$=_{def}$	$\mathscr{R}_1(M).\mathscr{R}_1(\sigma)$	$\mathscr{R}_1(\sigma \supset \tau)$	$=_{def}$	$\mathscr{R}_1(\sigma) \supset \mathscr{R}_1(\tau)$
$\mathscr{R}_1(\lambda x : \sigma.M)$	$=_{def}$	$\lambda(\mathscr{R}_1(\sigma), \mathscr{R}_1(M))$	$\mathscr{R}_1(\forall t.\sigma)$	$=_{def}$	$\Lambda\mathscr{R}_1(\sigma)$
$\mathscr{R}_1(\Lambda t.M)$	$=_{def}$	$\Lambda\mathscr{R}_1(M)$			

Lemma 5.1 The ES reduction system is strongly normalizing on $\mathscr{RT}$.

Proof. One shows that if A, B are raw terms in $\mathscr{RT}$ such that $A \rightarrow_{ES} B$, we have $\mathscr{R}_1(A) \succ_{\mathscr{S}} \mathscr{R}_1(B)$. □

Since the ES reduction system is also locally confluent, by Newman's Lemma we may conclude that it is confluent. Thus, if $A \in \mathscr{RT}$, we shall use $ES(A)$ to denote the unique ES-normal form of A. Since the ZT subcalculus is SN (it is included in ES) and locally confluent, by Newman's Lemma we may also conclude that it is confluent. Thus, if σ is

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

a type, we use $ZT(\sigma)$ to denote its unique ZT-normal form. Also, we will use $=_{ZT}$ to denote the reflexive, symmetrical and transitive closure of one step reduction in ZT.

Remark 5.2 One may verify in the style of Ríos (1993, pp. 63–64) that if σ is a type, then $ZT(\sigma)$ is a pure type, and if A is a raw term, $ES(A)$ is a pure term or type. Note that if $M \in \mathscr{T}$, then $ZT(M)$ not only may not be a pure term (for example if it has occurrences of the term substitution operator) but also may not have pure types, since term substitutions may block the execution of type substitutions.

Before proceeding to the SN property of $ES^{\bullet}$, we mention some technical properties of ZT and ES that will be used later.

Lemma 5.3 Let ρ, σ, τ be pure types, P and Q terms in $\mathscr{T}$, and τ_1, τ_2 any types.

1 $\rho\{t \leftarrow \tau\}\{u \leftarrow \sigma\{t \leftarrow \tau\}\} \equiv \rho\{u \leftarrow \sigma\}\{t \leftarrow \tau\}$ if $u \notin FTV(\tau)$.
2 $ZT(\tau_1[t := \tau_2]) \equiv ZT(\tau_1)\{t \leftarrow ZT(\tau_2)\}$.
3 $ES(P\langle x := Q\rangle) \equiv ES(P)\{x \leftarrow ES(Q)\}$.
4 $ES(P[t := \tau_1]) \equiv ES(P)\{t \leftarrow ZT(\tau_1)\}$.

Proof. The first item is proved by induction on ρ. The second item is first proved for τ_1 and τ_2 pure types and then using Remark 5.2. Item three is dealt with in a similar fashion. Item four is similar, but uses the observation that for a type σ we have $ES(\sigma) \equiv ZT(\sigma)$, and also uses item (2). □

Next we extend the SN property of ES to the reduction system $ES^{\bullet}$, that is, ES with the additional rule $\tau Beta$. For this we use Lemma A.1 (see the Appendix). We thus define a translation $\mathscr{R}_2(.)$ from terms in $\mathscr{RT}$ to $\mathscr{S}$ such that the conditions of Lemma A.1 are met.

Definition 5.6 We define the translation $\mathscr{R}_2(.) : \mathscr{RT} \longrightarrow \mathscr{S}$ as

$$\begin{array}{llllll}
\mathscr{R}_2(x) = \mathscr{R}_2(\sigma) & =_{def} & \star & \mathscr{R}_2(\Lambda t.M) & =_{def} & \Lambda\mathscr{R}_2(M) \\
\mathscr{R}_2(MN) & =_{def} & \mathscr{R}_2(M).\mathscr{R}_2(N) & \mathscr{R}_2(M\langle x := N\rangle) & =_{def} & \mathscr{R}_2(M)[\![\mathscr{R}_2(N)]\!] \\
\mathscr{R}_2(M\sigma) & =_{def} & \mathscr{R}_2(M) & \mathscr{R}_2(M[t := \tau]) & =_{def} & \mathscr{R}_2(M) \\
\mathscr{R}_2(\lambda x : \sigma.M) & =_{def} & \lambda(\mathscr{R}_2(\sigma), \mathscr{R}_2(M)). & & &
\end{array}$$

Lemma 5.4 Let M and N be terms in $\mathscr{T}$. Then:

1 If $M \rightarrow_{\tau Beta} N$, then $\mathscr{R}_2(M) \succ_{\mathscr{S}} \mathscr{R}_2(N)$.
2 If $M \rightarrow_{ES} N$, then $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(N)$.

Proof. In both items we use induction on the reduction $M \rightarrow_r N$ with $r \equiv \tau Beta$ or $r \in ES$. Let us consider the first item. Suppose the reduction is at the root. Then $M \equiv (\Lambda t.P)\tau$ and $N \equiv P[t := \tau]$, and $\mathscr{R}_2(M) \equiv \Lambda\mathscr{R}_2(P)$ and $\mathscr{R}_2(N) \equiv \mathscr{R}_2(P)$, and hence we obtain the desired result. If the reduction is internal, we apply the inductive hypothesis and the monotonicity of RPOs.

The second item is similar. Note that if $M \rightarrow_{ZT} N$ or $M \rightarrow_{zappt} N$, then $\mathscr{R}_2(M) \equiv \mathscr{R}_2(N)$. For the remaining rules we first consider the case where the reduction takes place at the

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

root, since, when the reduction is internal, we may apply the induction hypothesis and the monotonicity of $\succeq_{\mathscr{S}}$.

— $r = zapp$. Then $M \equiv (PQ)\langle x := R\rangle$ and $N \equiv P\langle x := R\rangle Q\langle x := R\rangle$ and $\mathscr{R}_2(M) \equiv (\mathscr{R}_2(P).\mathscr{R}_2(Q))[\![\mathscr{R}_2(R)]\!]$ and $\mathscr{R}_2(N) \equiv (\mathscr{R}_2(P)[\![\mathscr{R}_2(R)]\!].(\mathscr{R}_2(Q)[\![\mathscr{R}_2(R)]\!])$. Since $[\![]\!] \gg .$ we must verify that $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(P)[\![\mathscr{R}_2(R)]\!]$ and $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(Q)[\![\mathscr{R}_2(R)]\!]$. Let us consider the first of these, the other one being similar. By the 'equal heads' case, we have $\{\{\mathscr{R}_2(P).\mathscr{R}_2(Q), \mathscr{R}_2(R)\}\} \succeq'_{\mathscr{S}} \{\{\mathscr{R}_2(P), \mathscr{R}_2(R)\}\}$.

— $r = zlam$. Then $M \equiv (\lambda y : \sigma.P)\langle x := Q\rangle$ and $N \equiv \lambda y : \sigma.P\langle x := Q\rangle$, and we have $\mathscr{R}_2(M) \equiv (\lambda(\mathscr{R}_2(\sigma), \mathscr{R}_2(P))[\![\mathscr{R}_2(Q)]\!]$ and $\mathscr{R}_2(N) \equiv \lambda(\mathscr{R}_2(\sigma), \mathscr{R}_2(P)[\![\mathscr{R}_2(Q)]\!])$. Since $[\![]\!] \gg \lambda$, we must verify that $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(\sigma)$ and $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(P)[\![\mathscr{R}_2(Q)]\!]$. The first is direct; for the second we may verify that $\{\{\lambda(\mathscr{R}_2(\sigma), \mathscr{R}_2(P)), \mathscr{R}_2(Q)\}\} \succeq'_{\mathscr{S}} \{\{\mathscr{R}_2(P), \mathscr{R}_2(Q)\}\}$.

— $r = zvar, zgc$. The result follows by the subterm property of recursive path orderings.

— $r = zlamt$. Then $M \equiv (\Lambda t.P)\langle x := Q\rangle$ and $N \equiv \Lambda t.P\langle x := Q\rangle$ and $\mathscr{R}_2(M) \equiv (\Lambda\mathscr{R}_2(P))[\![\mathscr{R}_2(Q)]\!]$ and $\mathscr{R}_2(N) \equiv \Lambda(\mathscr{R}_2(P)[\![\mathscr{R}_2(Q)]\!])$. Then $[\![]\!] \gg \Lambda$ and we must thus verify that $\mathscr{R}_2(M) \succeq_{\mathscr{S}} \mathscr{R}_2(P)[\![\mathscr{R}_2(Q)]\!]$. We conclude using the 'equal heads' case. □

Corollary 5.5 The $ES^\bullet$ reduction system is strongly normalizing on $\mathscr{T}$.

Definition 5.7 (The z and λz reduction systems) Let us denote the reduction system obtained by taking ES and eliminating the subsystem ZT and rules *zappt* and *zlamt*, the z-reduction system. We define λz as z and the rewrite rule *Beta2*. We recall these rules below.

$$\lambda\mathtt{z}\begin{cases} (\lambda x : \sigma.M)N \to_{Beta2} M\langle x := N\rangle \\ \mathtt{z}\begin{cases} (MN)\langle x := P\rangle \to_{zapp} M\langle x := P\rangle N\langle x := P\rangle \\ (\lambda y : \sigma.M)\langle x := P\rangle \to_{zlam} \lambda y : \sigma.M\langle x := P\rangle \\ x\langle x := P\rangle \to_{zvar} P \\ M\langle x := P\rangle \to_{zgc} M \quad x \notin FV(M) \end{cases} \end{cases}$$

The λz and z calculi are the explicit substitution calculus for the simply typed lambda calculus (typed λx) and its subcalculus for computing explicit substitutions (typed x), respectively. Note that from Lemma 5.1 we have that z is strongly normalizing.

5.2. *The typed F-calculus with explicit substitutions*

In this section we introduce the typing rules for F_{es}. As already mentioned, in Bloo (1997) explicit substitutions were added to the Pure Type Systems formalism (Barendregt 1992). Although we deal only with an explicit substitution version of F, already for this calculus subject reduction was shown to fail. The counterexample exhibited in Bloo (1997) is the pure term $M \equiv (\Lambda s.(\lambda f : s \supset s.\lambda x : s.fx))t$. Due to lack of space we refer the reader interested in Explicit Pure Type Systems (EPTS) to Bloo (1997). After proving subject reduction for F_{es}, we conclude by showing how the typing rules deal with M. Independently, in Bloo (1999), the notion of Explicit Type System is introduced, which

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

is a presentation of pure type systems with explicit substitutions that enjoys the subject reduction property.

Definition 5.8 (Type assignment) A *type assignment* is a finite set of pairs $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ such that the variables are pairwise distinct and each σ_i is a (not necessarily pure) type. The domain of the type assignment is the set $\{x_1, \ldots, x_n\}$. If the types σ_i with $i \in 1..n$ are pure, then Γ is said to be a *pure type assignment*. Type assignments are denoted using capital greek letters $\Gamma, \Delta, \ldots$. We use $Dom(\Gamma)$ to denote the domain of Γ.

Definition 5.9 (Type judgement) A *type judgement* is an expression of the form $\Gamma \rhd M : \sigma$ where Γ is a type assignment, M is a term in $\mathscr{T}$ and σ is a *pure* type. If, moreover, Γ is a pure type assignment and M is a pure term, then $\Gamma \rhd M : \sigma$ is said to be a *pure type judgement*.

We shall sometimes use $\Gamma(x_i) = \sigma_i$ to denote the type assigned to the variable x_i by the type assignment $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$. Also, we use the abbreviation $t \notin FTV(\Gamma)$ for $t \notin FTV(\Gamma(y))$ for every $y \in Dom(\Gamma)$.

Definition 5.10 Let $\Gamma = \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ be a type assignment, t a type variable and τ a type. Then $\Gamma_{[t:=\tau]}$ is any member of the set of type assignments $\{\ \{x_1 : \sigma'_1, \ldots, x_n : \sigma'_n\}\ \mid \sigma'_i =_{ZT} \sigma_i[t := \tau]$ *for all* $i \in 1..n\}$.

Definition 5.11 (Derivable type judgements) The set of *derivable type judgements* in F_{es} is defined by the typing rules given in Figure 2. The rule *tabs* is subject to the following restriction: $t \notin FTV(\Gamma)$.

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : ZT(\sigma)}\ (var)$$

$$\frac{\Gamma \rhd M : \sigma \supset \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd MN : \tau}\ (app) \qquad \frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd (\lambda x : \sigma.M) : ZT(\sigma) \supset \tau}\ (abs)$$

$$\frac{\Gamma \rhd M : \forall t.\sigma}{\Gamma \rhd M\tau : \sigma\{t \leftarrow ZT(\tau)\}}\ (tapp) \qquad \frac{\Gamma \rhd M : \sigma}{\Gamma \rhd (\Lambda t.M) : \forall t.\sigma}\ (tabs)$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau \quad \Gamma \rhd N : ZT(\sigma)}{\Gamma \rhd M\langle x := N\rangle : \tau}\ (subs) \qquad \frac{\Gamma \rhd M : \sigma}{\Gamma_{[t:=\tau]} \rhd M[t := \tau] : \sigma\{t \leftarrow ZT(\tau)\}}\ (tsubs)$$

Fig. 2. Typing rules of the F_{es}-calculus

Definition 5.12 A term $M \in \mathscr{T}$ such that there exists a type assignment Γ and a pure type σ such that the judgement $\Gamma \rhd M : \sigma$ is derivable in the type system of Figure 2 is called *typable*. The set of typable terms is denoted $\mathscr{T}^{\rightarrow}$.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

In a typing judgement $\Gamma \rhd M : \sigma$ the type σ assigned to the term M is always a pure type. Also, we refer to the size of a derivation as the number of applications of rules it contains.

Example 5.6 Two examples of type derivations follow:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f : s \supset s, x : s \rhd f : s \supset s \quad f : s \supset s, x : s \rhd x : s}{f : s \supset s, x : s \rhd fx : s}\,(app)}{f : s \supset s \rhd \lambda x : s.fx : s \supset s}\,(abs)}{f : t \supset t \rhd (\lambda x : s.fx)[s := t] : t \supset t}\,(tsubs)}{\rhd(\lambda f : t \supset t.(\lambda x : s.fx)[s := t]) : (t \supset t) \supset (t \supset t)}\,(abs)}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f : s \supset s, x : s \rhd f : s \supset s \quad f : s \supset s, x : s \rhd x : s}{f : s \supset s, x : s \rhd fx : s}\,(app)}{f : s \supset s \rhd \lambda x : s.fx : s \supset s}\,(abs)}{f : (s \supset s)[s := t] \rhd (\lambda x : s.fx)[s := t] : t \supset t}\,(tsubs)}{\rhd \lambda f : (s \supset s)[s := t].(\lambda x : s.fx)[s := t] : (t \supset t) \supset (t \supset t)}\,(abs)}$$

Note that for the application of *tsubs* in the upper type derivation we use $(t \supset t) =_{ZT} (s \supset s)[s := t]$.

The typing rules of the usual (Church style presentation) of F is given by all the rules in Figure 2 except rules *subs* and *tsubs* and by restricting all type judgements to pure type judgements. Thus, when we wish to refer to derivable type judgements in F we shall always do so explicitly, otherwise we shall assume that the type system referred to is F_{es}.

In devising a rule for typing type substitutions our first intuition is that it should resemble an application of the rule for type variable abstraction followed by that of type variable application. But the introduction of a type substitution in a term M presents some difficulties. Consider a derivable premise $\Gamma \rhd M : \sigma$ and suppose that we add the type substitution $[t := \tau]$ to M in order to obtain $\Gamma \rhd M[t := \tau]$. Then we immediately note that the context Γ should not remain the same, since if $x \in FV(M)$, the type assigned to x by the type assignment Γ will have now fallen under the scope of the type substitution. For example, consider the application of *tsubs* in the upper derivation of Example 5.6. The variable f, which has type $s \supset s$ in the premise, has now fallen under the scope of type substitution $[s := t]$, so its type must be affected accordingly. This motivates the use of $\Gamma_{[t:=\tau]}$ in the rule *tsubs*.

An alternative approach could be to include the application of the type substitutions in the types of the variables in the type assignment Γ from the start. This requires some mecanism of control in order to ensure that the type substitution applied to the term M is the same as the one that has been applied to the type of the variable in the type assignment (and in the same order if more than one has been applied).

We shall first consider the relation between typing in F and typing in F_{es}. As expected, all terms typable in F are typable in F_{es} (the typing rules for F_{es} include those of F). And if a term is typable in F_{es}, then its ES normal form is typable in F (Lemma 5.9).

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

The following result is used when showing that F has the subject reduction property and will be used later.

Lemma 5.7 Let $\Gamma, x : \tau \rhd M : \sigma$ and $\Gamma \rhd N : \tau$ be derivable pure type judgements in F. Then $\Gamma \rhd M\{x \leftarrow N\} : \sigma$ is derivable in F.

Proof. The proof is by induction on the size of the derivation of $\Gamma, x : \tau \rhd M : \sigma$. □

Derivability is closed under type substitution in F.

Lemma 5.8 Let $\Gamma \rhd M : \sigma$ be a pure type judgement and τ a pure type. If $\Gamma \rhd M : \sigma$ is derivable in F and $BTV(M, \sigma, \Gamma) \cap FTV(\tau) = \varnothing$, then $\Gamma\{t \leftarrow \tau\} \rhd M\{t \leftarrow \tau\} : \sigma\{t \leftarrow \tau\}$ is derivable in F.

Proof. The proof is by induction on the size of the derivation of $\Gamma \rhd M : \sigma$. □

Note that the *subs* rule internalises Lemma 5.7 and rule *tsubs* internalises Lemma 5.8.

Lemma 5.9 Let $\Gamma \rhd M : \sigma$ be a derivable type judgement in F_{es}. Then $ZT(\Gamma) \rhd ES(M) : \sigma$ is derivable in F.

Proof. The proof is by induction on the derivation of $\Gamma \rhd M : \sigma$ using Lemmas 5.7 and 5.8 and Lemma 5.3 (2) and (3). We shall consider the interesting cases only:

— case (*subs*). The derivation runs

$$\frac{\Gamma, x : \rho \rhd P : \sigma \quad \Gamma \rhd Q : ZT(\rho)}{\Gamma \rhd P\langle x := Q\rangle : \sigma}\ (subs)$$

By the induction hypothesis, we have that $ZT(\Gamma), x : ZT(\rho) \rhd ES(P) : \sigma$ and $ZT(\Gamma) \rhd ES(Q) : ZT(\rho)$ are derivable type judgments in F. Then by Lemma 5.7, we have that $ZT(\Gamma) \rhd ES(P)\{x \leftarrow ES(Q)\} : \sigma$ is a derivable type judgement in F. Finally, applying Lemma 5.3(3), we are done.

— case (*tsubs*). The derivation runs

$$\frac{\Delta \rhd P : \rho}{\Delta_{[t:=\tau]} \rhd P[t := \tau] : \rho\{t \leftarrow ZT(\tau)\}}\ (tsubs)$$

where $\Gamma \equiv \Delta_{[t:=\tau]}$ and $\sigma \equiv \rho\{t \leftarrow ZT(\tau)\}$. Then by the induction hypothesis, the type judgement $ZT(\Delta) \rhd ES(P) : \rho$ is derivable in F. Applying Lemma 5.8, we obtain that $ZT(\Delta)\{t \leftarrow ZT(\tau)\} \rhd ES(P)\{t \leftarrow ZT(\tau)\} : \rho\{t \leftarrow ZT(\tau)\}$ is derivable in F. We conclude by observing the following:

- By Lemma 5.3(4), we have $ES(P[t := \tau]) \equiv ES(P)\{t \leftarrow ZT(\tau)\}$.
- On the other hand, suppose $x : \rho_1 \in \Delta$. Then we have $x : \rho_2 \in \Delta_{[t:=\tau]}$ with $\rho_2 =_{ZT} \rho_1[t := \tau]$. Since ZT is confluent and SN, we have $ZT(\rho_2) \equiv ZT(\rho_1[t := \tau]) \equiv_{L\ 5.3(2)} ZT(\rho_1)\{t \leftarrow ZT(\tau)\}$, which concludes the case since we then have $ZT(\Delta_{[t:=\tau]}) \equiv ZT(\Delta)\{t \leftarrow ZT(\tau)\}$.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

□

In order to prove the subject reduction property for F_{es} we need two auxiliary results: the weakening lemma and the context reduction lemma.

Lemma 5.10 (Weakening) If $\Gamma \rhd M : \sigma$ is a derivable type judgement, and $x : \tau$ is such that $x \notin Dom(\Gamma) \cup BV(M)$, and $BTV(M) \cap FTV(\tau) = \varnothing$, then $\Gamma, x : \tau \rhd M : \sigma$ is a derivable type judgement.

Proof. The proof is by induction on the size of the derivation of $\Gamma \rhd M : \sigma$. The interesting cases are:

— case (*abs*). Then the derivation runs

$$\frac{\Gamma, y : \rho \rhd P : \rho'}{\Gamma \rhd \lambda y : \rho.P : ZT(\rho) \supset \rho'}\,(abs)$$

Now by the conditions of the lemma we have $x \not\equiv y$ and thus by the induction hypothesis we obtain

$$\frac{\Gamma, y : \rho, x : \tau \rhd P : \rho'}{\Gamma, x : \tau \rhd \lambda y : \rho.P : ZT(\rho) \supset \rho'}\,(abs)$$

— case (*tsubs*). Then the derivation runs

$$\frac{\Delta \rhd P : \sigma'}{\Delta_{[t:=\rho]} \rhd P[t := \rho] : \sigma'\{t \leftarrow ZT(\rho)\}}\,(tsubs)$$

where $\Gamma \equiv \Delta_{[t:=\rho]}$ and $\sigma \equiv \sigma'\{t \leftarrow ZT(\rho)\}$. Then by the induction hypothesis we have

$$\frac{\Delta, x : \tau \rhd P : \sigma'}{\Delta_{[t:=\rho]}, x : \tau \rhd P[t := \rho] : \sigma'\{t \leftarrow ZT(\rho)\}}\,(tsubs)$$

Note that since $t \in BTV(P[t := \rho])$, by the condition of the lemma, we have $t \notin FTV(\tau)$ and therefore $\tau[t := \rho] \rightarrow_{ztgc} \tau$. □

We shall need the following lemma for the subject reduction property: it states that reduction in contexts conserves typability.

Lemma 5.11 (Context reduction) Let $\Gamma, x : \tau \rhd P : \sigma$ be any derivable type judgement and suppose $\tau \rightarrow_{ZT} \tau'$. Then $\Gamma, x : \tau' \rhd P : \sigma$ is a derivable type judgement.

Proof. The proof is by induction on the size of the derivation of $\Gamma, x : \tau \rhd P : \sigma$.

— case (*var*). We have two further subcases to consider:

– $P \equiv x$. Then $\sigma \equiv ZT(\tau)$. And since $\tau \rightarrow_{ZT} \tau'$ we have $ZT(\tau) \equiv ZT(\tau')$. Then $\Gamma, x : \tau' \rhd x : ZT(\tau)$ is a derivable type judgement.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

– $P \equiv y \not\equiv x$. Then $\sigma \equiv ZT(\rho)$ with $y : \rho \in \Gamma$, and we are done.

— cases (*app*), (*abs*), (*tapp*), (*subs*) and (*tabs*). We use the induction hypothesis. For the case (*tabs*) note that if $\tau \rightarrow_{ZT} \tau'$, then $FTV(\tau') \subseteq FTV(\tau)$.

— case (*tsubs*). Then the derivation runs

$$\frac{\Delta, x : \rho \rhd P' : \sigma_2}{\Delta_{[t:=\sigma_1]}, x : \tau \rhd P'[t := \sigma_1] : \sigma_2\{t \leftarrow ZT(\sigma_1)\}}\,(tsubs)$$

where $\tau =_{ZT} \rho[t := \sigma_1]$. But then $\tau' =_{ZT} \rho[t := \sigma_1]$. And we may construct the following derivation

$$\frac{\Delta, x : \rho \rhd P' : \sigma_2}{\Delta_{[t:=\sigma_1]}, x : \tau' \rhd P'[t := \sigma_1] : \sigma_2\{t \leftarrow ZT(\sigma_1)\}}\,(tsubs)$$

□

Lemma 5.12 (Subject reduction) Let $\Gamma \rhd M : \sigma$ be a derivable type judgement and suppose $M \rightarrow_{F_{es}} N$. Then $\Gamma \rhd N : \sigma$ is a derivable type judgement.

Proof. The proof is by induction on the reduction $M \rightarrow_{F_{es}} N$ using Lemmas 5.10 and 5.11 (see the Appendix). □

We finish this section with the counter-example (Bloo 1997, Proposition 7.27) mentioned in the introduction of the section, which states that the EPTS *Fx* does not verify the subject reduction property, and we show how this situation is remedied.

Example 5.13 Consider the pure term $M \equiv (\Lambda s.(\lambda f : s \supset s.\lambda x : s.fx))t$. It is typable in F_{es}

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f : s \supset s, x : s \rhd f : s \supset s \quad f : s \supset s, x : s \rhd x : s}{f : s \supset s, x : s \rhd fx : s}\,(app)}{f : s \supset s \rhd \lambda x : s.fx : s \supset s}\,(abs)}{\rhd \lambda f : s \supset s.\lambda x : s.fx : (s \supset s) \supset (s \supset s)}\,(abs)}{\rhd \Lambda s.(\lambda f : s \supset s.\lambda x : s.fx) : \forall s.(s \supset s) \supset (s \supset s)}\,(tabs)}{\rhd (\Lambda s.(\lambda f : s \supset s.\lambda x : s.fx))t : (t \supset t) \supset (t \supset t)}\,(tapp)$$

Now we have the following reduction sequence:

$$\begin{array}{lll} M & \rightarrow_{\tau Beta} & (\lambda f : s \supset s.\lambda x : s.fx)[s := t] \\ & \rightarrow_{ztlam} & \lambda f : (s \supset s)[s := t].(\lambda x : s.fx)[s := t] \\ & \overset{*}{\rightarrow}_{ZT} & \lambda f : t \supset t.(\lambda x : s.fx)[s := t] \end{array}$$

The last term is also typable, as illustrated by Example 5.6 (as well as the intermediate term $\lambda f : (s \supset s)[s := t].(\lambda x : s.fx)[s := t]$).

5.3. *Strong normalization*

In this subsection we prove the strong normalization property for the typed F_{es} calculus. We follow the presentation given in Gallier (1990). The idea is to define an erasing function *Erase*(.) that eliminates all typing information and proceed as follows:

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

1 Show that if a term resulting from erasing all type information, say $Erase(M)$, is λx-strongly normalizing, then the original term, that is, M, is F_{es}-strongly normalizing.
2 Thus, the result follows if we can show that $Erase(M)$ is λx-strongly normalizing for typable M. We generalize this goal by showing that if M is typable, then $Erase(M)$ concatenated by a series of explicit substitutions is λx-strongly normalizing; clearly this implies that $Erase(M)$ itself is λx-strongly normalizing.
3 This previous goal is proved by using the candidates of reducibility technique where the candidates are suitable subsets of $SN_{\lambda x}$.

Although we have chosen to give a direct proof of strong normalization of F_{es} by means of the reducibility technique, other proofs relating strong normalization of F_{es} to that of F are possible, as already mentioned in the introduction of this work.

We shall begin by considering the type information erasing function and then consider the reducibility proof.

5.3.1. *Basic definitions* The following erasing function removes all typing information from terms in $\mathscr{T}$.

Definition 5.13 (Erasing function) We define the function $Erase(.) : \mathscr{T} \longrightarrow \mathscr{T}_{\lambda x}$,

$$\begin{array}{llll} Erase(x) & =_{def} x & Erase(\Lambda t.M) & =_{def} Erase(M) \\ Erase(MN) & =_{def} Erase(M)Erase(N) & Erase(M\sigma) & =_{def} Erase(M) \\ Erase(\lambda x : \sigma.M) & =_{def} \lambda x.Erase(M) & Erase(M[t := \sigma]) & =_{def} Erase(M) \\ Erase(M\langle x := N\rangle) & =_{def} Erase(M)\langle x := Erase(N)\rangle & & \end{array}$$

We recall that x is the substitution subcalculus of λx. It propagates the substitution operators until they are performed or eliminated. The following lemma shows that a $\rightarrow_{F_{es}}$ reduction step may collapse to a λx-term via the erasing function or be simulated by a λx-reduction step.

Lemma 5.14 Let $M, N \in \mathscr{T}^{\rightarrow}$.

1 If $M \rightarrow_r N$ with $r \in (F_{es} \setminus \lambda z)$, then $Erase(M) \equiv Erase(N)$.
2 If $M \rightarrow_{\lambda z} N$, then $Erase(M) \rightarrow_{\lambda x} Erase(N)$.

Proof. Both items are proved by induction on the reduction $M \rightarrow_r N$ with $r \in (ES^{\bullet} \setminus z)$ and $r \in \lambda z$, respectively. We prove the second item:

— The reduction takes place at the root.
  – $r \equiv Beta2$. Then $M \equiv (\lambda x : \sigma.P)Q$ and $N \equiv P\langle x := Q\rangle$. And $Erase(M) \equiv (\lambda x.Erase(P))Erase(Q)$ and $Erase(N) \equiv Erase(P)\langle x := Erase(Q)\rangle$. And therefore $Erase(M) \rightarrow_{Beta} Erase(N)$.
  – $r \equiv zapp$. Then $M \equiv (PQ)\langle x := R\rangle$ and $N \equiv P\langle x := R\rangle Q\langle x := R\rangle$. And therefore on the one hand $Erase(M) \equiv (Erase(P)Erase(Q))\langle x := Erase(R)\rangle$, and on the other $Erase(N) \equiv Erase(P)\langle x := Erase(R)\rangle Erase(Q)\langle x := Erase(R)\rangle$, and by the rule *app* we are done.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

- $r \equiv zlam$. Then $M \equiv (\lambda y : \sigma.P)\langle x := Q\rangle$ and $N \equiv \lambda y : \sigma.P\langle x := Q\rangle$. And therefore $Erase(M) \equiv (\lambda y.Erase(P))\langle x := Erase(Q)\rangle$, and we also have $Erase(N) \equiv \lambda y.Erase(P)\langle x := Erase(Q)\rangle$, and by the rule *lam* we are done.
- $r \equiv var$ or $r \equiv gc$. We resolve these cases as above.

— The reduction is internal. In all cases we use the induction hypothesis and compatibility of $\rightarrow_{\lambda \mathtt{x}}$. Also note that in the cases where the context is $P\sigma$ or $P[t := \sigma]$ only reductions in P may have taken place. □

Proposition 5.15 Let M be a term in $\mathscr{T}^{\rightarrow}$. If there is an infinite $\rightarrow_{F_{es}}$-reduction sequence starting from M, then there is an infinite $\rightarrow_{\lambda \mathtt{x}}$-reduction sequence starting from $Erase(M)$.

Proof. Suppose we have an infinite $\rightarrow_{F_{es}}$-reduction sequence starting from M. Then, since the reduction system $S = F_{es} \setminus \lambda \mathtt{z}$ is strongly normalizing (as a consequence of Corollary 5.5), the reduction sequence must have the form $M \equiv M_1 \overset{*}{\rightarrow}_S M_2 \rightarrow_{\lambda \mathtt{z}} M_3 \overset{*}{\rightarrow}_S M_4 \rightarrow_{\lambda \mathtt{z}} M_5 \ldots$ where the reductions $\rightarrow_{\lambda \mathtt{z}}$ occur infinitely many times. Then we obtain, using Lemma 5.14, an infinite $\lambda \mathtt{x}$-reduction sequence starting from $Erase(M)$. □

Our aim in the next subsection will be to show that if M is a typable term in the polymorphic lambda calculus with explicit substitutions, then the untyped $\lambda \mathtt{x}$-term $Erase(M)$ is $\lambda \mathtt{x}$-strongly normalizing, thus allowing us to conclude that M is F_{es}-strongly normalizing. As already mentioned, we will use Tait's version of Girard's candidates of reducibility technique (Tait 1975).

5.3.2. *The proof of strong normalization of F_{es}* In this section we apply Tait's version of the technique of candidates of reducibility in which the candidates are untyped $\lambda \mathtt{x}$-terms.

Definition 5.14 (Substitution) A *substitution* is a function $\phi : \mathscr{X} \longrightarrow \mathscr{T}_{\lambda \mathtt{x}}$ such that $\phi(x) \neq x$ for only finitely many $x \in \mathscr{X}$. The finite set $\{x \mid \phi(x) \neq x\}$ is called the *domain* of ϕ and is denoted $Dom(\phi)$. When ϕ has domain $Dom(\Gamma)$ for some type assignment Γ of interest we write $\phi : \Gamma \longrightarrow \mathscr{T}_{\lambda \mathtt{x}}$.

Definition 5.15 (P-substitution) A substitution ϕ is called a *p(arallel)-substitution* if for every $x_i \in Dom(\phi)$ we have $x_i \notin \bigcup_{x \in Dom(\phi)} FV(\phi(x))$.

Definition 5.16 (Extension of a substitution) Let ϕ be a (possibly p-) substitution and let $[i_1, \ldots, i_n]$ be any ordering of the variables in $Dom(\phi)$. Then an *extension* $\overline{\phi}: \mathscr{T}_{\lambda \mathtt{x}} \longrightarrow \mathscr{T}_{\lambda \mathtt{x}}$ of ϕ is defined as $\overline{\phi}(M) =_{def} M\langle x_{i_1} := \phi(x_{i_1})\rangle \ldots \langle x_{i_n} := \phi(x_{i_n})\rangle$.

Intuitively, a p-substitution represents a parallel substitution. Usually when one defines a substitution ϕ and applies it to a term, the substitution process takes place 'in parallel'. Since $\lambda \mathtt{x}$ incorporates sequential substitution into the object level, and in order to apply the usual reducibility technique, we give conditions (definition of p-substitutions) on these sequential substitutions so that they may behave as desired.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Definition 5.17 Let C and D be sets of untyped terms in $\mathscr{T}_{\lambda \mathtt{x}}$. Then we define the *function space of C in D*, denoted $[C \to D]$, as $[C \to D] = \{M \in \mathscr{T}_{\lambda \mathtt{x}} \mid \forall N \in C,\ MN \in D\}$. We refer to $[. \to .]$ as the *function space constructor*.

Definition 5.18 A non-empty family of sets of (untyped) terms in $\mathscr{T}_{\lambda \mathtt{x}}$, $\mathscr{C}$, is called *type closed* if it verifies the following properties:

(i) every $C \in \mathscr{C}$ is non-empty;
(ii) $\mathscr{C}$ is closed under the function space constructor;
(iii) given any $\mathscr{C}$-indexed family $(A_C)_{C \in \mathscr{C}}$ of sets in $\mathscr{C}$, then $\bigcap_{C \in \mathscr{C}} A_C \in \mathscr{C}$; and
(iv) every $C \in \mathscr{C}$ is closed under α-equivalence.

We shall use the notation $M \in \bigcup \mathscr{C}$ to say that M is an untyped $\lambda \mathtt{x}$-term in C for some member C of the family $\mathscr{C}$.

Definition 5.19 (Assignment) Let $\mathscr{C}$ be a type closed family. An *assignment* is a function $\eta : \mathscr{V} \longrightarrow \mathscr{C}$. Given a set $C \in \mathscr{C}$ and a type variable t, we use $\eta[t := C]$ to denote the assignment such that for all $v \in \mathscr{V}$ we have $\eta[t := C](v) \equiv C$ if $v \equiv t$, and $\eta[t := C](v) \equiv \eta(v)$ otherwise. Although the $.[. := .]$ symbol has already been used for the closure operator, we do not expect the overloading to cause any confusion.

Definition 5.20 Given an assignment $\eta : \mathscr{V} \longrightarrow \mathscr{C}$, for every pure type σ, the set $[\![\sigma]\!]\eta$ is defined as

$$\begin{array}{lll} [\![t]\!]\eta & =_{def} & \eta(t) \text{ if } t \in \mathscr{V} \\ [\![(\sigma_1 \supset \sigma_2)]\!]\eta & =_{def} & [[\![\sigma_1]\!]\eta \to [\![\sigma_2]\!]\eta] \\ [\![\forall t.\sigma]\!]\eta & =_{def} & \bigcap_{C \in \mathscr{C}} [\![\sigma]\!]\eta[t := C] \end{array}$$

In the following lemma we assume by the variable convention that the bound type variables in σ do not occur free in τ. The third item is referred to as 'Girard's trick' (Gallier 1990). The proof may be found in Gallier (1990).

Lemma 5.16

1 Given two assignments $\eta_1 : \mathscr{V} \longrightarrow \mathscr{C}$ and $\eta_2 : \mathscr{V} \longrightarrow \mathscr{C}$, for every pure type σ, if η_1 and η_2 agree on $FTV(\sigma)$, then $[\![\sigma]\!]\eta_1 = [\![\sigma]\!]\eta_2$.
2 Let σ and τ be pure types. Then for every assignment $\eta : \mathscr{V} \longrightarrow \mathscr{C}$ we have $[\![\sigma\{t \leftarrow \tau\}]\!]\eta = [\![\sigma]\!]\eta[t := [\![\tau]\!]\eta]$.
3 Let $\mathscr{C}$ be a type closed family. Then for every assignment $\eta : \mathscr{V} \longrightarrow \mathscr{C}$, for every pure type σ, we have $[\![\sigma]\!]\eta \in \mathscr{C}$.

In order to prove strong normalization on typable terms, we impose some conditions on the members of a type closed family. Before proceeding we recall some notation: if $M, N_1, \ldots, N_m \in \mathscr{T}_{\lambda \mathtt{x}}$ and $x_1, \ldots, x_m \in \mathscr{X}$, we use $M^\star$ to denote the term $M\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle$ for $m \geqslant 0$.

Definition 5.21 We say that a family $\mathscr{C}$ of sets of untyped $\lambda \mathtt{x}$-terms is a *family of candidates of reducibility* iff it is type closed and satisfies the following. For every set $C \in \mathscr{C}$ we have:

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

R1. For every variable $x \in \mathscr{X}$, $x \in C$.
R2. For all $M, N \in \mathscr{T}_{\lambda x}$, if $M\langle x := N\rangle \in C$, then $(\lambda x.M)\ N \in C$.
R3. For all $M \in \bigcup \mathscr{C}$ and for all $m \geqslant 0$, and $N_1, \ldots, N_m \in \bigcup \mathscr{C}$ if $M^\star \in C$, then $x\langle x := M\rangle^\star \in C$
R4. For all $M, N \in \bigcup \mathscr{C}$ and for every $x \in \mathscr{X}$ and all $m \geqslant 0$ and $N_1, \ldots, N_m \in \bigcup \mathscr{C}$ if $M^\star \in C$ and $x \notin FV(M)$, then $M\langle x := N\rangle^\star \in C$.
R5. For all $M_1, M_2 \in \mathscr{T}_{\lambda x}$, for every $x \in \mathscr{X}$ and all $m \geqslant 0$, and $N, N_1, \ldots, N_m \in \bigcup \mathscr{C}$ if $(M_1\langle x := N\rangle M_2\langle x := N\rangle)^\star \in C$, then $(M_1 M_2)\langle x := N\rangle^\star \in C$.
R6. For all $M \in \mathscr{T}_{\lambda x}$, for every $x \in \mathscr{X}$, and all $m \geqslant 0$, and $N, N_1, \ldots, N_m \in \bigcup \mathscr{C}$, if $(\lambda y.M\langle x := N\rangle)^\star \in C$ and $y \notin FV(N)$, then $(\lambda y.M)\langle x := N\rangle^\star \in C$.
R7. For all $M, N \in \mathscr{T}_{\lambda x}$, for all $m \geqslant 0$, for all $P, N_1, \ldots, N_m \in \bigcup \mathscr{C}$, if $y \notin FV(P)$ and $M\langle x := P\rangle\langle y := N\langle x := P\rangle\rangle^\star \in C$, then $M\langle y := N\rangle\langle x := P\rangle^\star \in C$.

Condition (R7) is the explicit version of the substitution lemma (Lemma 2.2).

Now we can prove (roughly) that if a term in $\mathscr{T}$ is typable with type σ, its image via any extension of a p-substitution ϕ (satisfying the conditions of the lemma) is in the member set of the family $\mathscr{C}$ interpreting the type σ.

Lemma 5.17 Let $\mathscr{C}$ be a family of candidates of reducibility. For every derivation of $\Gamma \rhd M : \sigma$ for some term $M \in \mathscr{T}$, for every assignment $\eta : \mathscr{V} \longrightarrow \mathscr{C}$, for every p-substitution $\phi : \Gamma \longrightarrow \mathscr{T}_{\lambda x}$, if $\phi(x) \in [\![ZT(\Gamma(x))]\!]\eta$ for every $x \in Dom(\Gamma)$, then for every extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(M)) \in [\![\sigma]\!]\eta$.

Proof. The proof is by induction on the size of the derivation of $\Gamma \rhd M : \sigma$ (see appendix). □

In order to obtain families of candidates of reducibility for proving properties of $Erase(M)$ where $M \in \mathscr{T}$, Girard has shown that conditions R1–R7 must be strengthened.

Definition 5.22 (Saturated set)

(i) Let S be a non-empty set of terms in $\mathscr{T}_{\lambda x}$. We say that S is *closed* iff whenever $Mx \in S$, we have $M \in S$ where $x \in \mathscr{X}$.
(ii) A subset $C \subseteq S$ that is closed under α-equivalence is called *saturated* iff the following conditions hold:

S1. For every variable $x \in \mathscr{X}$, for all $n \geqslant 0$ and all $P_1 \ldots P_n \in S$, $xP_1 \ldots P_n \in C$.
S2. For all $M, N \in \mathscr{T}_{\lambda x}$, all $n \geqslant 0$ and all $P_1, \ldots, P_n \in S$, if $M\langle x := N\rangle P_1 \ldots P_n \in C$, then $(\lambda x.M)\ NP_1 \ldots P_n \in C$.
S3. For all $M \in S$ and for all $m \geqslant 0, n \geqslant 0$ and all $N_1, \ldots, N_m \in S$ and $P_1, \ldots, P_n \in S$ if $M^\star P_1 \ldots P_n \in C$, then $x\langle x := M\rangle^\star P_1 \ldots P_n \in C$.
S4. For all $M, N \in S$, all $m \geqslant 0$, $n \geqslant 0$, all $N_1, \ldots, N_m$, all $P_1, \ldots, P_n \in S$, and every variable x in $\mathscr{X}$ such that $x \notin FV(M)$ if $M^\star P_1 \ldots P_n \in C$, then $M\langle x := N\rangle^\star P_1 \ldots P_n \in C$.
S5. For all $M_1, M_2 \in \mathscr{T}_{\lambda x}$, for every variable $x \in \mathscr{X}$ and for all $m \geqslant 0$, and $N, N_1, \ldots, N_m \in S$ and all $n \geqslant 0$ and $P_1, \ldots, P_n \in S$ if $(M_1\langle x := N\rangle M_2\langle x := N\rangle)^\star P_1 \ldots P_n \in C$, then $(M_1 M_2)\langle x := N\rangle^\star P_1 \ldots P_n \in C$.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

S6. For all $M \in \mathscr{T}_{\lambda \mathtt{x}}$, for every $x \in \mathscr{X}$, and all $m \geqslant 0$, and $N, N_1, \ldots, N_m \in S$, and all $n \geqslant 0$ and $P_1, \ldots, P_n \in S$ if $(\lambda y.M\langle x := N\rangle)^\star P_1 \ldots P_n \in C$, then $(\lambda y.M)\langle x := N\rangle^\star P_1 \ldots P_n \in C$.

S7. For all $M, N \in \mathscr{T}_{\lambda \mathtt{x}}$, for all $m \geqslant 0$, for all $P, N_1, \ldots, N_m \in S$, for all $n \geqslant 0$ and $P_1, \ldots, P_n \in S$ if $y \notin FV(P)$ and $M\langle x := P\rangle\langle y := N\langle x := P\rangle\rangle^\star P_1 \ldots P_n \in C$, then $M\langle y := N\rangle\langle x := P\rangle^\star P_1 \ldots P_n \in C$.

Lemma 5.18 Let S be a non-empty closed set of terms in $\mathscr{T}_{\lambda \mathtt{x}}$, let $\mathscr{C}$ be the family of saturated subsets of S, and assume that S is saturated. Then $\mathscr{C}$ is a family of candidates of reducibility.

Proof. Note that conditions (S1)–(S7) imply conditions (R1)–(R7), respectively. Therefore we are left to verify that $\mathscr{C}$ is a type closed family. First of all note that the family $\mathscr{C}$ is non-empty, since $S \in \mathscr{C}$. Also,

— By condition (S1) each saturated subset is non-empty, since it contains all variables.
— Let C and D be saturated subsets of S. We must show that $[C \to D] = \{M \in \mathscr{T}_{\lambda \mathtt{x}} \mid \forall N \in C,\ MN \in D\}$ is a saturated subset of S.
  - $[C \to D]$ is a subset of S. Let $M \in [C \to D]$. Since there is some variable $x \in C$, we have $Mx \in D$. And since S is closed, $M \in S$.
  - (S1). Since D is a saturated subset of S, by (S1) for every variable $x \in \mathscr{X}$ and for all $n \geqslant 0$ and $P_1, \ldots, P_n, P \in S$, we have $xP_1 \ldots P_nP \in D$. Since this is valid for every $P \in C$, we have $xP_1 \ldots P_n \in [C \to D]$.
  - (S2). Let M and N be any terms in $\mathscr{T}_{\lambda \mathtt{x}}$. Let $P_1 \ldots P_n$ be terms in S. Suppose that $M\langle x := N\rangle P_1 \ldots P_n \in [C \to D]$. Then for every $P \in C$ we have $M\langle x := N\rangle P_1 \ldots P_nP \in D$. Since D is saturated, by (S2) we have $(\lambda x.M)\ NP_1 \ldots P_nP \in D$. Since this holds for every $P \in C$, we have $(\lambda x.M)\ NP_1 \ldots P_n \in [C \to D]$.
  - (S3). Let M be a term in S. Let $N_1, \ldots, N_m$ and $P_1 \ldots P_n$ be terms in S. Suppose that $M\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle P_1 \ldots P_n \in [C \to D]$. Then for every $P \in C$ we have $M\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle P_1 \ldots P_nP \in D$, and since D is a saturated subset of S, by (S3), $x\langle x := M\rangle\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle P_1 \ldots P_nP \in D$. Since this holds for every $P \in C$, we have $x\langle x := M\rangle\langle x_1 := N_1\rangle \ldots \langle x_m := N_m\rangle P_1 \ldots P_n \in [C \to D]$.

  The remaining cases are dealt with likewise.
— Note that properties (S1)–(S7) of saturated subsets of S are preserved under arbitrary intersections, and thus for the $\mathscr{C}$-indexed family of saturated subsets of S, $(A_C)_{C \in \mathscr{C}}$, we have $\bigcap_{C \in \mathscr{C}} A_C$ is a saturated subset of S.
— A saturated subset of S is closed under α-equivalence by definition. □

Proposition 5.19 Let S be a non-empty closed set of terms in $\mathscr{T}_{\lambda \mathtt{x}}$, let $\mathscr{C}$ be the family of all saturated subsets of S, and assume that S is saturated. Let $M \in \mathscr{T}$ that type checks under the type assignment $\Gamma \equiv \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$, and let $z_1, \ldots, z_n$ be fresh variables. Then $Erase(M)\langle x_{i_1} := z_1\rangle \ldots \langle x_{i_n} := z_n\rangle \in S$ where $i_1, \ldots, i_n$ is any ordering of the variables in $Dom(\Gamma)$.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Proof. By Lemma 5.18 we have that $\mathscr{C}$ is a family of candidates of reducibility. Finally, we conclude by Lemma 5.16(3) and Lemma 5.17. □

Therefore in order to prove properties of $Erase(M)$ when M is a typable polymorphic term, we need saturated sets satisfying these properties. In particular, since we are interested in λx- strongly normalizing terms ($SN_{\lambda x}$), we shall prove that the set $SN_{\lambda x}$ is a closed saturated set. The key ingredient in the proof of this result is the Perpetuality Proposition.

Lemma 5.20 The set $SN_{\lambda x}$ is a closed saturated set.

Proof. We must verify that $SN_{\lambda x}$ is closed and verifies (S1)–(S7). Note that if Mx is λx-strongly normalizable, then M also, so $SN_{\lambda x}$ is closed. We are left to check that $SN_{\lambda x}$ verifies

— (S1) Suppose $P_1 \dots P_n \in SN_{\lambda x}$, then for every variable $x \in \mathscr{X}$, $xP_1 \dots P_n$ is λx- strongly normalizing.
— (S2) By the Perpetuality Proposition 3.10(1).
— (S3)-(S6) By the Perpetuality Proposition 3.10(2),(3).
— (S7) By Remark 3.11. □

Therefore, since $SN_{\lambda x}$ is a non-empty closed subset of $\mathscr{T}_{\lambda x}$ and is itself saturated, if we construct the family of saturated subsets of $SN_{\lambda x}$, we obtain, by Lemma 5.18, a family of candidates of reducibility. This allows us to prove the following corollary.

Corollary 5.21 Let M be a term in $\mathscr{T}^{\rightarrow}$. Then M is strongly $\rightarrow_{F_{es}}$-normalizing.

Proof. Suppose $M \in \mathscr{T}^{\rightarrow}$. Then there is a type assignment $\Gamma \equiv \{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$ and a pure type σ such that $\Gamma \rhd M : \sigma$ is a derivable type judgement.

Let $z_1, \dots, z_n$ be fresh variables. Then by Proposition 5.19, we have $Erase(M)\langle x_{i_1} := z_1\rangle \dots \langle x_{i_n} := z_n\rangle \in SN_{\lambda x}$ where $i_1, \dots, i_n$ is any ordering of the variables in $Dom(\Gamma)$. But then we also have $Erase(M) \in SN_{\lambda x}$, and, finally, by Proposition 5.15, we may conclude that M is F_{es}-strongly normalizing. □

6. Conclusions

We have studied perpetuality in the calculus of explicit substitutions λx. As mentioned in the introduction, current results on perpetuality for reduction systems require orthogonality, a property that the λx-calculus does not fulfill. Thus a different treatment is needed, a task that we have undertaken in the present work.

As applications of the perpetuality studies, we have: an inductive characterization of the λx-strongly normalizing terms; the formulation of perpetual reduction strategies for λx; and an application to a reducibility proof of strong normalization for a polymorphic lambda calculus with explicit substitutions F_{es}. We have thus introduced the polymorphic lambda calculus with explicit substitutions F_{es} and have shown that it satisfies the subject reduction property and that all typable terms are strongly normalizing. In Bloo (1997),

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Pure Type Systems with explicit substitutions (EPTS) were defined and subject reduction was already shown to fail for $F\mathtt{x}$ (the explicit substitution version of F).

Extending the reducibility technique to other calculi of explicit substitution is an interesting goal. Recent work in this direction is E. Ritter's article Ritter (1999). There, the reducibility technique is used for showing preservation of strong normalization (PSN) of explicit substitution calculi. As regards the present work, the reducibility has been used to prove SN of a typed calculus and in a polymorphic setting. An extension of the present work to other calculi of explicit substitution will be considered.

Also, the extension of the perpetuality analysis to calculi of explicit substitution that include a composition operator, such as $\lambda\sigma$, should be considered. This could lead to the study of results such as standardization and developments in explicit substitution calculi (Melliès 1996), as has been done for the λ-calculus (Barendregt 1984).

Note that from the results presented above we can obtain a simplified variant of Benaissa *et al.*'s minimal derivation technique for proving PSN (Benaissa *et al.* 1996). First of all, no commutation lemma is needed. As we have already mentioned, commutation results are highly technical and very tedious. Also, note that there was no need for the definition of internal and external positions, and its corresponding reduction. The notion of void reduction suffices. This could permit the study of a general schema for proving the PSN property of calculi of explicit substitution, just as the schema presented in Kesner (1996) for (ground) confluence of calculi of explicit substitution.

Appendix A.

Definition A.1 (RPO) Let $s = f(s_1, .., s_m)$ and $t = g(t_1, .., t_n)$ be terms in $\mathscr{T}_l$. Then $s \succ_{\mathscr{S}} t$ if and only if one of the following holds:

1 (subterm) $s_i \succ_{\mathscr{S}} t$ or $s_i = t$ for some $i \in 1..m$
2 (decreasing heads) $f \gg g$ and $s \succ_{\mathscr{S}} t_1, .., s \succ_{\mathscr{S}} t_n$
3 (equal heads) $f = g$ and $\{\!\{s_1, .., s_m\}\!\} \succ'_{\mathscr{S}} \{\!\{t_1, .., t_n\}\!\}$

where $\succ'_{\mathscr{S}}$ is the extension of $\succ_{\mathscr{S}}$ to multisets.

Lemma A.1 Let $A = (S, R_1 \cup R_2)$ be an abstract reduction system and $>$ be a well-founded order on a set T such that:

1 R_2 is strongly normalizing.
2 There exists a function $h : S \to T$ such that $a \to_{R_1} b$ implies $h(a) > h(b)$ and $a \to_{R_2} b$ implies $h(a) \geqslant h(b)$.

Then $R_1 \cup R_2$ is strongly normalizing.

Lemma A.2 (Subject reduction) Let $\Gamma \rhd M : \sigma$ be a derivable type judgement and suppose $M \to_{F_{es}} N$. Then $\Gamma \rhd N : \sigma$ is a derivable type judgement.

Proof. The proof is by induction on the reduction $M \to_r N$ with $r \in F_{es}$. Thus we consider the cases when the reduction takes place at the root and when the reduction is internal.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Suppose the reduction takes place at the root. Then we have the following cases to analyse:

— $r \equiv Beta2$. Then $M \equiv (\lambda x : \tau.P)Q$ and $N \equiv P\langle x := Q\rangle$ and the derivation runs

$$\dfrac{\dfrac{\Gamma, x : \tau \rhd P : \sigma}{\Gamma \rhd \lambda x : \tau.P : ZT(\tau) \supset \sigma}(abs) \qquad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd (\lambda x : \tau.P)Q : \sigma}(app)$$

and we obtain

$$\dfrac{\Gamma, x : \tau \rhd P : \sigma \quad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd P\langle x := Q\rangle : \sigma}(subs)$$

— $r \equiv \tau Beta$. Then $M \equiv (\Lambda t.P)\tau$ and $N \equiv P[t := \tau]$ and the derivation runs

$$\dfrac{\dfrac{\Gamma \rhd P : \sigma'}{\Gamma \rhd \Lambda t.P : \forall t.\sigma'}(tabs)}{\Gamma \rhd (\Lambda t.P)\tau : \sigma'\{t \leftarrow ZT(\tau)\}}(tapp)$$

where $t \notin FTV(\Gamma)$. We may obtain

$$\dfrac{\Gamma \rhd P : \sigma'}{\Gamma \rhd P[t := \tau] : \sigma'\{t \leftarrow ZT(\tau)\}}(tsubs)$$

Note that $\Gamma \in \{\ \{x_1 : \sigma'_1, \ldots, x_n : \sigma'_n\} \ \mid\ \sigma'_i =_{ZT} \sigma_i[t := \tau]$ *for all* $i \in 1..n\}$ where $\Gamma = \{\sigma_1, \ldots, \sigma_n\}$, since $t \notin FTV(\Gamma)$.

— $r \equiv zapp$. Then $M \equiv (PQ)\langle x := R\rangle$ and $R \equiv P\langle x := R\rangle Q\langle x := R\rangle$ and the derivation runs as follows

$$\dfrac{\dfrac{\Gamma, x : \tau \rhd P : \sigma_1 \supset \sigma \quad \Gamma, x : \tau \rhd Q : \sigma_1}{\Gamma, x : \tau \rhd PQ : \sigma}(app) \qquad \Gamma \rhd R : ZT(\tau)}{\Gamma \rhd (PQ)\langle x := R\rangle : \sigma}(subs)$$

and we may obtain

$$\dfrac{\dfrac{\Gamma, x : \tau \rhd P : \sigma_1 \supset \sigma \quad \Gamma \rhd R : ZT(\tau)}{\Gamma \rhd P\langle x := R\rangle : \sigma_1 \supset \sigma}(subs) \qquad \dfrac{\Gamma, x : \tau \rhd Q : \sigma_1 \quad \Gamma \rhd R : ZT(\tau)}{\Gamma \rhd Q\langle x := R\rangle : \sigma_1}(subs)}{\Gamma \rhd P\langle x := R\rangle Q\langle x := R\rangle : \sigma}(app)$$

— $r \equiv zlam$. Then $M \equiv (\lambda y : \sigma_1.P)\langle x := Q\rangle$ and $N \equiv \lambda y : \sigma_1.P\langle x := Q\rangle$. The derivation runs

$$\dfrac{\dfrac{\Gamma, x : \tau, y : \sigma_1 \rhd P : \sigma_2}{\Gamma, x : \tau \rhd \lambda y : \sigma_1.P : ZT(\sigma_1) \supset \sigma_2}(abs) \qquad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd (\lambda y : \sigma_1.P)\langle x := Q\rangle : ZT(\sigma_1) \supset \sigma_2}(subs)$$

Now, by the variable convention we may assume that all bound type variables in Q do not occur free in σ_1 and that y does not occur bound in Q. Then by Lemma 5.10 we have that $\Gamma, y : \sigma_1 \rhd Q : \tau$ is a derivable type judgement. This allows us to construct the following derivation:

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

$$\dfrac{\dfrac{\Gamma, x:\tau, y:\sigma_1 \rhd P : \sigma_2 \quad \Gamma, y:\sigma_1 \rhd Q : ZT(\tau)}{\Gamma, y:\sigma_1 \rhd P\langle x := Q\rangle : \sigma_2}\,(subs)}{\Gamma \rhd \lambda y:\sigma_1.P\langle x := Q\rangle : ZT(\sigma_1) \supset \sigma_2}\,(abs)$$

— $r \equiv zvar$. Then $M \equiv x\langle x := P\rangle$, $N \equiv P$, $\sigma \equiv ZT(\sigma')$ and the derivation runs

$$\dfrac{\Gamma, x:\sigma' \rhd x : ZT(\sigma') \quad \Gamma \rhd P : ZT(\sigma')}{\Gamma \rhd x\langle x := P\rangle : ZT(\sigma')}\,(subs)$$

And the subderivation ending in the $\Gamma \rhd P : ZT(\sigma')$ suffices.

— $r \equiv zgc$. Then $M \equiv Q\langle x := P\rangle$ and $N \equiv Q$ with $x \notin FV(Q)$, and the derivation runs

$$\dfrac{\Gamma, x:\tau \rhd Q : ZT(\sigma') \quad \Gamma \rhd P : ZT(\tau)}{\Gamma \rhd Q\langle x := P\rangle : ZT(\sigma')}\,(subs)$$

But one may verify by induction that if $\Gamma, x:\tau \rhd Q : \sigma$ is a derivable type judgement such that $x \notin FV(Q)$, then $\Gamma \rhd Q : \sigma$ is also a derivable type judgement. This concludes this case.

— $r \equiv zappt$. Then $M \equiv (P\tau')\langle x := Q\rangle$ and $N \equiv P\langle x := Q\rangle\tau'$ and $\sigma \equiv \sigma'\{t \leftarrow ZT(\tau')\}$ where

$$\dfrac{\dfrac{\Gamma, x:\tau \rhd P : \forall t.\sigma'}{\Gamma, x:\tau \rhd P\tau' : \sigma'\{t \leftarrow ZT(\tau')\}}\,(tapp) \qquad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd (P\tau')\langle x := Q\rangle : \sigma'\{t \leftarrow ZT(\tau')\}}\,(subs)$$

Then we may commute the application of rules *tapp* and *subs* as follows

$$\dfrac{\dfrac{\Gamma, x:\tau \rhd P : \forall t.\sigma' \quad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd P\langle x := Q\rangle : \forall t.\sigma'}\,(subs)}{\Gamma \rhd P\langle x := Q\rangle\tau' : \sigma'\{t \leftarrow ZT(\tau')\}}\,(tapp)$$

— $r \equiv zlamt$. Then $M \equiv (\Lambda t.P)\langle x := Q\rangle$ and $N \equiv \Lambda t.P\langle x := Q\rangle$ and the derivation runs

$$\dfrac{\dfrac{\Gamma, x:\tau \rhd P : \rho}{\Gamma, x:\tau \rhd \Lambda t.P : \forall t.\rho}\,(tabs) \qquad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd (\Lambda t.P)\langle x := Q\rangle : \forall t.\rho}\,(subs)$$

where $t \notin FTV(\Gamma, x:\tau)$. And we may obtain

$$\dfrac{\dfrac{\Gamma, x:\tau \rhd P : \rho \quad \Gamma \rhd Q : ZT(\tau)}{\Gamma \rhd P\langle x := Q\rangle : \rho}\,(subs)}{\Gamma \rhd \Lambda t.P\langle x := Q\rangle : \forall t.\rho}\,(tabs)$$

— $r \equiv ztapp$. Then $M \equiv (PQ)[t := \tau]$ and $N \equiv P[t := \tau]Q[t := \tau]$ and the derivation runs

$$\dfrac{\dfrac{\Delta \rhd P : \sigma_1 \supset \sigma_2 \quad \Delta \rhd Q : \sigma_1}{\Delta \rhd PQ : \sigma_2}\,(app)}{\Delta_{[t:=\tau]} \rhd (PQ)[t := \tau] : \sigma_2\{t \leftarrow ZT(\tau)\}}\,(tsubs)$$

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

And we may obtain

$$\dfrac{\dfrac{\Delta \rhd P : \sigma_1 \supset \sigma_2}{\Delta_{[t:=\tau]} \rhd P[t := \tau] : (\sigma_1 \supset \sigma_2)\{t \leftarrow ZT(\tau)\}}(tsubs) \quad \dfrac{\Delta \rhd Q : \sigma_1}{\Delta_{[t:=\tau]} \rhd Q[t := \tau] : \sigma_1\{t \leftarrow ZT(\tau)\}}(tsubs)}{\Delta_{[t:=\tau]} \rhd P[t := \tau]Q[t := \tau] : \sigma_2\{t \leftarrow ZT(\tau)\}}(app)$$

— $r \equiv ztlam$. Then $M \equiv (\lambda y : \sigma'.P)[t := \tau]$ and $N \equiv \lambda y : \sigma'[t := \tau].P[t := \tau]$ and the derivation runs

$$\dfrac{\dfrac{\Delta, y : \sigma' \rhd P : \rho}{\Delta \rhd \lambda y : \sigma'.P : ZT(\sigma') \supset \rho}(abs)}{\Delta_{[t:=\tau]} \rhd (\lambda y : \sigma'.P)[t := \tau] : (ZT(\sigma') \supset \rho)\{t \leftarrow ZT(\tau)\}}(tsubs)$$

and we may obtain

$$\dfrac{\dfrac{\Delta, y : \sigma' \rhd P : \rho}{\Delta_{[t:=\tau]}, y : \sigma'[t := \tau] \rhd P[t := \tau] : \rho\{t \leftarrow ZT(\tau)\}}(tsubs)}{\Delta_{[t:=\tau]} \rhd \lambda y : \sigma'[t := \tau].P[t := \tau] : ZT(\sigma'[t := \tau]) \supset \rho\{t \leftarrow ZT(\tau)\}}(tsubs)$$

Then, by Lemma 5.3(2) we are done.

— $r \equiv ztvar2$. Then $M \equiv x[t := \tau]$ and $N \equiv x$ and the derivation runs

$$\dfrac{\Delta \rhd x : ZT(\rho)}{\Delta_{[t:=\tau]} \rhd x[t := \tau] : ZT(\rho)\{t \leftarrow ZT(\tau)\}}(tsubs)$$

where $x : \rho \in \Delta$. Let $x : \rho' \in \Delta_{[t:=\tau]}$ and therefore $\rho' =_{ZT} \rho[t := \tau]$. Then $ZT(\rho') \equiv ZT(\rho)\{t \leftarrow ZT(\tau)\}$ using Lemma 5.3(2) and the fact that ZT is complete. Then $\Delta_{[t:=\tau]} \rhd x : ZT(\rho') \equiv ZT(\rho)\{t \leftarrow ZT(\tau)\}$ is a derivable type judgement (using *var*). Note that if the *ztvar*2-rule were replaced by the *ztgc*2-rule, the following result, which may be proved by induction on the size of the derivation, would be required: if $\Delta \rhd M : \rho$ is derivable and $t \notin FTV(M)$, then $\Delta_{[t:=\tau]} \rhd M : \rho\{t \leftarrow ZT(\tau)\}$ is derivable for any type τ.

— $r \equiv ztappt$. Then $M \equiv (P\tau)[t := \tau']$ and $N \equiv P[t := \tau']\tau[t := \tau']$ and the derivation runs

$$\dfrac{\dfrac{\Delta \rhd P : \forall u.\rho}{\Delta \rhd P\tau : \rho\{u \leftarrow ZT(\tau)\}}(tapp)}{\Delta_{[t:=\tau']} \rhd (P\tau)[t := \tau'] : \rho\{u \leftarrow ZT(\tau)\}\{t \leftarrow ZT(\tau')\}}(tsubs)$$

and we may obtain

$$\dfrac{\dfrac{\Delta \rhd P : \forall u.\rho}{\Delta_{[t:=\tau']} \rhd P[t := \tau'] : (\forall u.\rho)\{t \leftarrow ZT(\tau')\} \equiv \forall u.\rho\{r \leftarrow ZT(\tau')\}}(tsubs)}{\Delta_{[t:=\tau']} \rhd P[t := \tau']\tau[t := \tau'] : \rho\{t \leftarrow ZT(\tau')\}\{u \leftarrow ZT(\tau[t := \tau'])\}}(tapp)$$

Then, using Lemma 5.3(1) and (2), we are done.

— $r \equiv ztlamt$. Then $M \equiv (\Lambda u.P)[t := \tau]$ and $N \equiv \Lambda u.P[t := \tau]$, and we assume by the variable convention that $u \notin FTV(\tau)$. Then the derivation runs

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

$$\dfrac{\dfrac{\Delta \rhd P : \rho}{\Delta \rhd \Lambda u.P : \forall u.\rho}\,(tabs)}{\Delta_{[t:=\tau]} \rhd (\Lambda u.P)[t := \tau] : (\forall u.\rho)\{t \leftarrow ZT(\tau)\}}\,(tsubs)$$

where $u \notin FTV(\Delta)$. We assume, by the variable convention, that $u \notin FTV(\Delta_{[t:=\tau]})$, and thus obtain the derivation

$$\dfrac{\dfrac{\Delta \rhd P : \rho}{\Delta_{[t:=\tau]} \rhd P[t := \tau] : \rho\{t \leftarrow ZT(\tau)\}}\,(tsubs)}{\Delta_{[t:=\tau]} \rhd \Lambda u.P[t := \tau] : \forall u.\rho\{t \leftarrow ZT(\tau)\}}\,(tabs)$$

Suppose now that the reduction is internal. Then we consider the following cases according to each possible context C:

— $C \equiv \lambda x : \tau.P$. We must consider two possible cases:
  – $P \rightarrow_{F_{es}} P'$ and therefore $N \equiv \lambda x : \tau.P'$. This case is handled by the induction hypothesis.
  – $\tau \rightarrow_{ZT} \tau'$ and therefore $N \equiv \lambda x : \tau'.P$. Then the derivation runs

$$\dfrac{\Gamma, x : \tau \rhd P : \rho}{\Gamma \rhd \lambda x : \tau.P : ZT(\tau) \supset \rho}\,(abs)$$

  And we may conclude by using the context reduction Lemma 5.11 on the derivation ending in the premise of *abs* followed by a new application of *abs*, and noting that $ZT(\tau) \equiv ZT(\tau')$.

— $C \equiv \Lambda t.P$. Then $P \rightarrow_{F_{es}} P'$ and $N \equiv \Lambda t.P'$, and we use the induction hypothesis.

— $C \equiv PQ$. Then either $P \rightarrow_{F_{es}} P'$, and therefore $N \equiv P'Q$, or $Q \rightarrow_{F_{es}} Q'$, and therefore $N \equiv PQ'$. In both cases we use the induction hypothesis.

— $C \equiv P\rho$. Then either $P \rightarrow_{F_{es}} P'$, and therefore $N \equiv P'\rho$, or $\rho \rightarrow_{ZT} \rho'$, and therefore $N \equiv P\rho'$. For the first case we use the induction hypothesis. The second case is resolved by a new application of *tapp* and by noting that $ZT(\rho) \equiv ZT(\rho')$.

— $C \equiv P\langle x := Q\rangle$. Then either $P \rightarrow_{F_{es}} P'$, and therefore $N \equiv P'\langle x := Q\rangle$, or $Q \rightarrow_{F_{es}} Q'$, and therefore $N \equiv P\langle x := Q'\rangle$. In both cases we use the induction hypothesis.

— $C \equiv P[t := \tau]$. Then the derivation runs

$$\dfrac{\Delta \rhd P : \rho}{\Delta_{[t:=\tau]} \rhd P[t := \tau] : \rho\{t \leftarrow ZT(\tau)\}}\,(tsubs)$$

and either $P \rightarrow_{F_{es}} P'$, and therefore $N \equiv P'[t := \tau]$, or $\tau \rightarrow_{ZT} \tau'$, and therefore $N \equiv P[t := \tau']$. For the first case we use the induction hypothesis. For the second we construct the following derivation.

$$\dfrac{\Delta \rhd P : \rho}{\Delta_{[t:=\tau]} \rhd P[t := \tau'] : \rho\{t \leftarrow ZT(\tau')\}}\,(tsubs)$$

We are left to verify that the assignment $\Delta_{[t:=\tau]}$ is in the set of type assignments $\{\ \{x_1 : \sigma'_1, \ldots, x_n : \sigma'_n\} \ \mid\ \sigma'_i =_{ZT} \sigma_i[t := \tau']\ \textit{for all}\ i \in 1..n\} = \Delta_{[t:=\tau']}$, where $\Delta \equiv \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$, which is true since $\tau \rightarrow_{ZT} \tau'$. □

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Lemma A.3 Let $\mathscr{C}$ be a family of candidates of reducibility. For every derivation of $\Gamma \rhd M : \sigma$ for some term $M \in \mathscr{T}$, for every assignment $\eta : \mathscr{V} \longrightarrow \mathscr{C}$, for every p-substitution $\phi : \Gamma \longrightarrow \mathscr{T}_{\lambda x}$, if $\phi(x) \in [\![ZT(\Gamma(x))]\!]\eta$ for every $x \in Dom(\Gamma)$, then for every extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(M)) \in [\![\sigma]\!]\eta$.

Proof. The proof is by induction on the size of the derivation of $\Gamma \rhd M : \sigma$.

— Base case. Here the derivation consists solely of $\Gamma \rhd x : ZT(\rho)$ where $x : \rho \in \Gamma$. Now, by hypothesis we know that $\phi(x) \in [\![ZT(\rho)]\!]\eta$. Let $[x_1, \ldots, x_n]$ be any ordering of the variables in $Dom(\Gamma)$ and suppose $x_j \equiv x$. Then we reason as follows:

$\phi(x) \in [\![ZT(\rho)]\!]\eta$	hypothesis
$\phi(x)\langle x_n := \phi(x_n)\rangle \in [\![ZT(\rho)]\!]\eta$	(R4), ϕ p- subs
.	.
.	.
$\phi(x)\langle x_{j+1} := \phi(x_{j+1})\rangle \ldots \langle x_n := \phi(x_n)\rangle \in [\![ZT(\rho)]\!]\eta$	(R4), ϕ p-subs
$x\langle x := \phi(x)\rangle\langle x_{j+1} := \phi(x_{j+1})\rangle \ldots \langle x_n := \phi(x_n)\rangle \in [\![ZT(\rho)]\!]\eta$	(R3)
.	.
.	.
$x\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle \in [\![ZT(\rho)]\!]\eta$	(R4), Γ type judg.

— Inductive case. Let $k+1$ be the length of this derivation. We analyse the last derivation rule applied:

– *app.* Thus the derivation ends as follows,

$$\frac{\Gamma \rhd M : \tau \supset \sigma \quad \Gamma \rhd N : \tau}{\Gamma \rhd MN : \sigma}\,(app)$$

Then, by the induction hypothesis, we have that $\overline{\phi}(Erase(M)) \in [\![\tau \supset \sigma]\!]\eta$, and also that $\overline{\phi}(Erase(N)) \in [\![\tau]\!]\eta$ for any extension $\overline{\phi}$ of ϕ. Then, by the definition of $[\![\tau \supset \sigma]\!]\eta$, we have $\overline{\phi}(Erase(M))\overline{\phi}(Erase(N)) \in [\![\sigma]\!]\eta$. That is,

$$Erase(M)\langle x_1 := \phi(x_1)\rangle..\langle x_n := \phi(x_n)\rangle Erase(N)\langle x_1 := \phi(x_1)\rangle..\langle x_n := \phi(x_n)\rangle \in [\![\sigma]\!]\eta.$$

Then, applying condition (R5) repeatedly, we obtain

$$\begin{aligned}(Erase(M)Erase(N))\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle & \\ \equiv \quad Erase(MN)\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle & \\ \in \quad [\![\sigma]\!]\eta &\end{aligned}$$

Since this applies for any extension $\overline{\phi}$ of ϕ meeting the conditions of the lemma, we are done.

– *abs.* Then the derivation ends as follows:

$$\frac{\Gamma, x : \tau \rhd M : \rho}{\Gamma \rhd \lambda x : \tau.M : ZT(\tau) \supset \rho}\,(abs)$$

Let N be any term in $[\![ZT(\tau)]\!]\eta$. Let $\overline{\phi}$ be any extension of ϕ with ordering of variables $[x_1, \ldots, x_n]$. We choose a representative in the α-equivalence

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

class of $[(\lambda x.Erase(M))\langle x_1 := \phi(x_1)\rangle \dots \langle x_n := \phi(x_n)\rangle]_\alpha$, say $(\lambda x'.Erase(M'))\langle x'_1 := \phi(x_1)\rangle \dots \langle x'_n := \phi(x_n)\rangle$ (note that renaming commutes with the erasing function) such that

1 $x' \notin FV(\phi(x_i))$ for $i \in 1..n$ and $x' \notin FV(N)$ and $x_i \not\equiv x'$ for $i \in 1..n$, and

2 $x'_i \notin FV(\phi(x_j))$ and $x'_i \notin FV(N)$, for $i, j \in 1..n$.

We have to rename x in order to avoid name clashes (first item). But since we are working with p-substitutions, this entails more renaming (second item) so that we may be able to apply the induction hypothesis when it is needed. Note that since ϕ is a p-substitution no renaming takes place inside the $\phi(x_i)$s in the representative of the α-equivalence class chosen.

Let ϕ' be the resulting p-substitution, that is, the substitution defined as

$$\phi'(z) =_{def} \begin{cases} \phi(x_i) & if\ z \equiv x'_i \\ N & if\ z \equiv x' \end{cases}$$

Since the induction hypothesis holds for every derivation of length $\leqslant k$ and for every assignment η satisfying the conditions of the lemma, we have that for every extension $\overline{\phi'}$ of ϕ', $\overline{\phi'}(Erase(M')) \in [\![\rho]\!]\eta$. In particular, $Erase(M')\langle x'_1 := \phi(x_1)\rangle \dots \langle x'_n := \phi(x_n)\rangle\langle x' := N\rangle \in [\![\rho]\!]\eta$.

Then, by condition (R2), we obtain that $(\lambda x'.Erase(M')\langle x'_1 := \phi(x_1)\rangle \dots \langle x'_n := \phi(x_n)\rangle)N \in [\![\rho]\!]\eta$

Since this is valid for every $N \in [\![ZT(\tau)]\!]\eta$, by definition of $[\![ZT(\tau) \supset \rho]\!]\eta$, we have $\lambda x'.Erase(M')\langle x'_1 := \phi(x_1)\rangle \dots \langle x'_n := \phi(x_n)\rangle \in [\![ZT(\tau) \supset \rho]\!]\eta$

Finally, since $x' \notin FV(\phi(x_i))$ for $i \in 1..n$ (first item above), we may apply condition (R6) repeatedly to obtain $(\lambda x'.Erase(M'))\langle x'_1 := \phi(x_1)\rangle \dots \langle x'_n := \phi(x_n)\rangle \in [\![ZT(\tau) \supset \rho]\!]\eta$

And, by closure under α-equivalence, we conclude

$$\begin{aligned} (\lambda x.Erase(M))\langle x_1 := \phi(x_1)\rangle \dots \langle x_n := \phi(x_n)\rangle & \\ \equiv\ & Erase(\lambda x : \tau.M)\langle x_1 := \phi(x_1)\rangle \dots \langle x_n := \phi(x_n)\rangle \\ \equiv\ & \overline{\phi}(Erase(\lambda x : \tau.M)) \\ \in\ & [\![ZT(\tau) \supset \rho]\!]\eta \end{aligned}$$

– *subs*. Then the derivation ends as follows

$$\frac{\Gamma, x : \tau \rhd M_1 : \sigma \quad \Gamma \rhd M_2 : ZT(\tau)}{\Gamma \rhd M_1\langle x := M_2\rangle : \sigma}\ (subs)$$

Let ϕ be any p-substitution with domain Γ satisfying the conditions of the lemma. Now we choose a representative of the equivalence class

$$[Erase(M_1)\langle x := Erase(M_2)\langle x_1 := \phi(x_1)\rangle \dots \langle x_n := \phi(x_n)\rangle\rangle]_\alpha$$

say $Erase(M'_1)\langle x' := Erase(M_2)\langle x_1 := \phi(x_1)\rangle \dots \langle x_n := \phi(x_n)\rangle\rangle$, such that

1 $x' \notin Dom(\Gamma)$, and

2 $x' \notin FV(\overline{\phi}(Erase(M_2))$ (and thus $x' \notin FV(\phi(x_j))$ with $j \in 1..n$).

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Let ϕ' be the substitution $\phi' : (\Gamma, x' : \tau) \longrightarrow \mathscr{T}_{\lambda x}$ defined as

$$\phi'(z) =_{def} \begin{cases} \phi(x_i) & if\ z \equiv x_i \\ \overline{\phi}(Erase(M_2)) & if\ z \equiv x' \end{cases}$$

Note that due to the first and second item, and because $x_i \notin FV(\overline{\phi}(Erase(M_2))$ with $i \in 1..n$ (since they are all bound and ϕ is a p-substitution), the resulting substitution is a p-substitution.
Since the induction hypothesis holds for any derivation of length $\leqslant k$ and for every assignment satisfying the conditions of the lemma, on the one hand for every extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(M_2)) \in [\![ZT(\tau)]\!]\eta$. Another application of the induction hypothesis to the derivation ending in the left premise of the application of *subs* allows us to infer that for every extension $\overline{\phi'}$ of ϕ' we have $\overline{\phi'}(Erase(M_1')) \in [\![\sigma]\!]\eta$. In particular we have,
$Erase(M_1')\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle\langle x' := Erase(M_2)\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle\rangle \in [\![\sigma]\!]\eta$.
Now, since $x' \notin FV(\phi(x_i))$ (item 2), we may apply condition (R7) repeatedly and obtain $Erase(M_1')\langle x' := Erase(M_2)\rangle\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle \in [\![\sigma]\!]\eta$.
And, since $[\![\sigma]\!]\eta$ is closed under α-equivalence,

$$\begin{aligned} Erase(M_1)\langle x := Erase(M_2)\rangle\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle & \\ \equiv\ & Erase(M_1\langle x := M_2\rangle)\langle x_1 := \phi(x_1)\rangle \ldots \langle x_n := \phi(x_n)\rangle \\ \equiv\ & \overline{\phi}(Erase(M_1\langle x := M_2\rangle)) \in [\![\sigma]\!]\eta \end{aligned}$$

– *tapp*. Then the derivation ends as follows:

$$\frac{\Gamma \rhd M : \forall t.\rho}{\Gamma \rhd M\tau : \rho\{t \leftarrow ZT(\tau)\}}\ (tapp)$$

By the induction hypothesis, for any extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(M)) \in [\![\forall t.\rho]\!]\eta$. Now, since $[\![\forall t.\rho]\!]\eta = \bigcap_{C \in \mathscr{C}} [\![\rho]\!]\eta[t := C]$, we have $\overline{\phi}(Erase(M)) \in [\![\rho]\!]\eta[t := C]$ for every $C \in \mathscr{C}$. In particular, if we take $C = [\![ZT(\tau)]\!]\eta \in \mathscr{C}$, we have $\overline{\phi}(Erase(M)) \in [\![\rho]\!]\eta[t := [\![ZT(\tau)]\!]\eta]$. By Lemma 5.16(2), we have that $[\![\rho\{t \leftarrow ZT(\tau)\}]\!]\eta = [\![\rho]\!]\eta[t := [\![ZT(\tau)]\!]\eta]$, and, therefore, $\overline{\phi}(Erase(M\tau)) = \overline{\phi}(Erase(M)) \in [\![\rho\{t \leftarrow ZT(\tau)\}]\!]\eta$.

– *tabs*. The derivation ends as follows:

$$\frac{\Gamma \rhd M : \rho}{\Gamma \rhd (\Lambda t.M) : \forall t.\rho}\ (tabs)$$

where $t \notin FTV(\Gamma(x))$ with $x \in Dom(\Gamma)$.
Then $t \notin FTV(ZT(\Gamma(x)))$ for every $x \in Dom(\Gamma)$ (because, if $\tau \rightarrow_{ZT} \tau'$, we have $FTV(\tau') \subseteq FTV(\tau)$) and, by Lemma 5.16(1), we have $[\![ZT(\Gamma(x))]\!]\eta = [\![ZT(\Gamma(x))]\!]\eta[t := C]$ for every $C \in \mathscr{C}$. Since the induction hypothesis holds for every derivation of length $\leqslant k$, for every η, and for every p-substitution ϕ satisfying the conditions of the lemma, it holds for every $C \in \mathscr{C}$ when applied to the derivation $\Gamma \rhd M : \rho$, to every $\eta[t := C]$ and to every p- sub-

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

stitution ϕ such that $\phi(x) \in [\![ZT(\Gamma(x))]\!]\eta$. Therefore, for every $C \in \mathscr{C}$ and for every extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(M)) \in [\![\rho]\!]\eta[t := C]$. And thus $\overline{\phi}(Erase(\Lambda t.M)) \equiv \overline{\phi}(Erase(M)) \in \bigcap_{C\in\mathscr{C}}[\![\rho]\!]\eta[t := C] \equiv [\![\forall t.\rho]\!]\eta$.

– *tsubs.* The derivation ends as follows:

$$\frac{\Delta \rhd P : \rho}{\Delta_{[t:=\tau]} \rhd P[t := \tau] : \rho\{t \leftarrow ZT(\tau)\}}\ (tsubs)$$

Let η be any assignment and ϕ be any p-substitution with domain $\Delta_{[t:=\tau]}$ meeting the conditions of the lemma. Consider any $x : \sigma_1 \in \Delta_{[t:=\tau]}$ and its corresponding $x : \sigma_2 \in \Delta$. Then, by hypothesis, we have $\phi(x) \in [\![ZT(\sigma_1)]\!]\eta$, and since $\sigma_1 =_{ZT} \sigma_2[t := \tau]$, we have $ZT(\sigma_1) \equiv ZT(\sigma_2)\{t \leftarrow ZT(\tau)\}$, using the fact that ZT is complete and Lemma 5.3(2). Thus $[\![ZT(\sigma_1)]\!]\eta = [\![ZT(\sigma_2)\{t \leftarrow ZT(\tau)\}]\!]\eta =_{L\ 5.16(2)} [\![ZT(\sigma_2)]\!]\eta[t := [\![ZT(\tau)]\!]\eta]$.
We then apply the induction hypothesis with p-substitution ϕ and assignment $\eta[t := [\![ZT(\tau)]\!]\eta]$ and obtain that for every extension $\overline{\phi}$ of ϕ we have $\overline{\phi}(Erase(P)) \in [\![\rho]\!]\eta[t := [\![ZT(\tau)]\!]\eta]$. But, since $Erase(P) \equiv Erase(P[t := \tau])$, and by Lemmas 5.16(2) and 5.3(2), we may conclude that $Erase(P[t := \tau]) \in [\![\rho\{t \leftarrow ZT(\tau)\}]\!]\eta$.

□

Acknowledgements.

I am grateful to Delia Kesner and Alejandro Ríos for valuable discussions and encouragement, to Roel Bloo for comments, to Pierre Lescanne for spotting some bugs in an earlier version of this work and to the anonymous referees.

References

Abadi, M., Cardelli, L., Curien, P.-L. and Lévy, J.-J. (1991) Explicit Substitutions. *Journal of Functional Programming* **4** (1) 375-416.

Barendregt, H. P. (1984) *The Lambda Calculus: its Syntax and Semantics* (revised edition), Studies in Logic and the Foundations of Mathematics **103**, North-Holland.

Barendregt, H. P. (1992) Lambda Calculi with Types. In: Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science, Volume 2 – Background: Computational Structures*, Oxford University Press.

Barendregt, H. P., Bergstra, J. A., Klop, J. W. and Volken, H. (1976) Degrees, reductions and representability in the lambda calculus. Technical report, Utrecht University, 1976, Preprint no. 22.

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

Bergstra, J. A. and Klop, J. W. (1982) Strong normalization and perpetual reductions in the lambda calculus. *Journal of Information Processing and Cybernetics* **18** 403–417.

Bloo, R. (1997) *Preservation of Termination for Explicit Substitution,* Ph. D. Thesis, Eindhoven University.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

Bloo, R. (1999) Pure Type Systems and Explicit Substitution. In: *Proceedings of the International Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs, Trento, Italy*.

Bloo, R. and Geuvers, H. (1998) Explicit Substitution: on the edge of strong normalization. *Theoretical Computer Science* **204**.

Bonelli, E. (1999) The polymorphic lambda calculus with explicit substitutions. In: *Proceedings of the International Workshop on Explicit Substitutions: Theory and Applications to Programs and Proofs, Trento, Italy*.

Church, A. and Rosser, J. B. (1936) Some properties of conversion. *Transactions of the American Mathematical Society* **39** 472–482.

Dershowitz, N. (1982) Orderings for term rewriting systems. *Theoretical Computer Science* **17** (3) 279–301.

Ferreira, M., Kesner, D. and Puel, L. (1996) Lambda-calculi with explicit substitutions and composition which preserve beta-strong normalization. Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP'96). *Springer-Verlag Lecture Notes in Computer Science* **1139**.

Gallier, J. H. (1990) On Girard's 'candidates de reducibilité'. In: Odifreddi, P. (ed.) *Logic in Computer Science*, Academic Press, APIC Series **31** 123–203.

Girard, J.-Y. (1972) *Interprétation Fonctionnelle et Elimination des Coupures de l'Arithmétique d'Ordre Supérieur,* Thèse d'Etat, Université Paris VII.

Girard, J.-Y., Lafont, Y. and Taylor, P. (1989) *Proofs and Types,* Cambridge University Press.

Hardin, T. and Lévy, J-J. (1989) A confluent calculus of substitutions. In: *France–Japan Artificial Intelligence and Computer Science Symposium.*

Kamareddine, F. and Ríos, A. (1995) A lambda calculus a la de Bruijn with Explicit Substitutions. Proceedings of the 7th International Symposium on Programming Languages: Implementations, Logics and Programs (PLILP'95). *Springer-Verlag Lecture Notes in Computer Science* **982**.

Kesner, D. (1996) Confluence properties of extensional and non-extensional lambda calculi with explicit substitutions. Proceedings of the 7th International Conference on Rewriting Techniques and Applications (RTA'96). *Springer-Verlag Lecture Notes in Computer Science* **1103**.

Khasidashvili, Z. (1994) The longest perpetual reductions in orthogonal expression reduction systems. In Proceedings of the 3rd. International Conference on Logical Foundations of Computer Science (LFCS'94). *Springer-Verlag Lecture Notes in Computer Science* **813**.

Khasidashvili, Z., Ogawa, M. and van Oostrom, V. (1999) Perpetuality and Uniform Normalization in Orthogonal Rewrite Systems. Accepted for publication in *Information and Computation.*

Klop, J. W. (1980) *Combinatory Reduction Systems,* Mathematical Centre Tracts **127**, CWI, Amsterdam.

Klop, J. W. (1992) Term rewriting systems. In: *Handbook of Logic in Computer Science, Volume 2*, Oxford University Press.

Lescanne, P. (1994) From λ_σ to λ_υ, a journey through calculi of explicit substitutions. In: *Ann. ACM Symp. on Principles of Programming Languages (POPL)*, ACM 60–69.

Melliès, P. A. (1995) Typed λ-calculi with explicit substitutions may not terminate. In: TLCA'95. *Springer-Verlag Lecture Notes in Computer Science* **902**.

Melliès, P. A. (1996) *Description Abstraite des Systèmes de Réécriture*, Ph.D. thesis, Université Paris VII.

Muñoz, C. A. (1996a) Confluence and Preservation of Strong Normalisation in an Explicit Substitutions Calculus. *Proceedings of the Eleven Annual IEEE Symposium on Logic in Computer Science.*

Muñoz, C. A. (1996b) Dependent Types with Explicit Substitutions: A meta-theoretical development.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press

In: Proceedings of the International Workshop TYPES'96. *Springer-Verlag Lecture Notes in Computer Science* **1512**.

Muñoz, C. A. (1997) A Left-Linear Variant of $\lambda\sigma$. In: *Proceedings of the International Conference PLILP/ALP/HOA'97, Southampton, UK*.

van Raamsdonk, F. and Severi, P. (1995) On normalization. Technical Report CS-R9545, CWI.

van Raamsdonk, F., Severi, P., Sørensen, M. H. and Xi, H. (1999) Perpetuality in the λ-calculus. *Journal of Information and Computation* **149** (2).

Reynolds, J. C. (1974) Towards a Theory of Type Structure. In: Robinet, B. (ed.) Programming Symposium, Proceedings, Colloque sur la Programmation. *Springer-Verlag Lecture Notes in Computer Science* **19**.

Ríos, A. (1993) *Contribution à l'étude des λ-calculs avec substitutions explicites,* Ph. D. Thesis, Université de Paris VII.

Ritter, E. (1999) Characterising Explicit Subsitutions which Preserve Termination (Extended Abstract). In: *Typed Lambda Calculi and Applications.*

Rose, K. (1992) Explicit Cyclic Substitutions. In: CTRS'92. *Springer-Verlag Lecture Notes in Computer Science* **656**.

Sørensen, M. H. (1996) Effective longest and infinite reduction paths in untyped λ-calculus. In: Kirchner, H. (ed.) Proc. of the 19th Internatinal Colloqium on Trees in Algebra and Programming, CAAP'96. *Springer-Verlag Lecture Notes in Computer Science* **1059**.

Tait, W. W. (1975) A realizability interpretation of the theory of species. In: Parikh, R. (ed.) Logic Colloquium. *Springer-Verlag Lecture Notes in Computer Science* **453**.

Xi, H. (1996) An induction measure on λ-terms and its applications. Research Report 96–192, Department of Mathematical Sciences, Carnegie Mellon University.

https://doi.org/10.1017/S0960129500003248 Published online by Cambridge University Press