JFP **14** (5): 519–546, September 2004.
DOI: 10.1017/S0956796803004970 Printed in the United Kingdom

Linearization of the lambda-calculus and its relation with intersection type systems

MÁRIO FLORIDO and LUÍS DAMAS

Department of Computer Science, Faculty of Science & LIACC, University of Porto, R. do Campo Alegre 823, 4150-180 Porto, Portugal
(*e-mail:* {amf,luis}@ncc.up.pt)

Abstract

In this paper we present a notion of expansion of a term in the lambda-calculus which transforms terms into linear terms. This transformation replaces each occurrence of a variable in the original term by a fresh variable taking into account non-trivial implications in the structure of the term caused by these simple replacements. We prove that the class of terms which can be expanded is the same of terms typable in an Intersection Type System, i.e. the strongly normalizable terms. We then show that expansion is preserved by weak-head reduction, the reduction considered by functional programming languages.

Capsule Review

A lambda term is linear if any bound or free type variable occurs at least once in it. Linear lambda terms are strongly normalizable and typable in Curry' simple type system, and their implementation can support various optimization techniques. This paper presents a methodology based on intersection types to translate any strongly normalizable lambda term in a linear one. The translation function (called Expansion in the paper) preserve meaning in the sense that it commutes with head reduction for arbitrary terms and with general beta reduction for terms of the lambda-l-calculus. Expansion is defined via the typing of a term in the full intersection type system and is thereof undecidable. However, combined with some proper decidable restriction of the intersection type system, it could provide a basis for some program optimization techniques.

1 Introduction

Linear programs are simple. Concerning implementation issues, for linear programs one may safely inline the term bound to any variable, or safely update structures in place. Type inference for linear programs is easy and has nice properties which do not hold in general. In this paper we show that linear programs (linear λ-terms) are enough to fully characterize terms typable by Intersection Types. By linear λ-terms we mean terms where every λ-abstraction binds at most one variable occurrence.

Intersection types originate in the works of Coppo & Dezani-Ciancaglini (1980) and Barendregt *et al.* (1983). ITSs without the universal type ω, as presented in Coppo & Dezani-Ciancaglini (1980), give a characterization of strongly normalizable terms, in the sense that a term is typed in an intersection type system without ω

if and only if it is strongly normalizable. These systems type more terms than the Curry type system (Curry, 1934) or the type system of pure ML (Damas & Milner, 1982). New attention was given to intersection type systems due to a result of Kfoury & Wells (1982), which proved that these systems are decidable for restrictions of finite rank, which correspond to a large class of typable terms. Although Kfoury and Wells's work made the technology of intersection types more accessible avoiding the use of the complicated notion of type expansion, intersection type systems are rather complex systems whose definitions can be quite difficult to understand.

ITSs are based on the idea of explicitly typing with a different type each different occurrence of a variable. In this way they are capable of typing terms where sharing of variables with non-unifiable types may cause non-typification in other type systems. For example, in ITS $\lambda x.xx$ has type $(\alpha \cap \alpha \to \beta) \to \beta$. A more interesting example is the term $T \equiv (\lambda x.xx)I$, where I is the identity function $\lambda x.x$. This term has type $\alpha \to \alpha$ which does not involve intersections, although it is not typable in the Curry Type System, because it has a non-typable subterm. However notice that there is a linear term, $(\lambda x_1 x_2.x_1 x_2)II$, which has the same type in the Curry Type System, and which is a linear version of $(\lambda x.xx)I$ in the sense that each occurrence of a shared variable in $(\lambda x.xx)I$ corresponds to a distinct variable in $(\lambda x_1 x_2.x_1 x_2)II$.

In this paper, we address the following problem: to which extent can we approximate a typed term in the intersection type system by a linear term?

The first difficulty in answering this question is what is meant by *linear version* of a term. Consider the following example: Suppose one wants to define the linear version of the term $(\lambda xy.xy)(\lambda z.zz)(\lambda x.x)$. In this term the only variable which occurs more than once is z. Thus we can expand $(\lambda z.zz)$ to get the term $(\lambda z_1 z_2.z_1 z_2)$. But $(\lambda z.zz)$ will have y as an argument after one reduction step. Thus y in $(\lambda xy.xy)$ has to be copied and we get the term

$$(\lambda xy.xyy)(\lambda z_1 z_2.z_2 z_2)(\lambda x.x)$$

Now, a variable which occurred once in the original term occurs twice in the expanded term, thus the expansion process has to go on, expanding $(\lambda xy.xyy)$ to $(\lambda xy_1 y_2.xy_1 y_2)$. Notice that y will be replaced by $\lambda x.x$ in the original version of the term, thus, as y was expanded to two new variables we have to duplicate $(\lambda x.x)$ in the expanded term to get the final term:

$$(\lambda xy_1 y_2.xy_1 y_2)(\lambda z_1 z_2.z_1 z_2)(\lambda x.x)(\lambda x.x)$$

The main contributions of this paper are the following:

- A notion of *term expansion*. Under that notion we show that one can define the *linear version* of a term if and only if the term is typable in an intersection type system which types exactly the strongly normalizable terms.
- To show that every term typable in an intersection type system has an expansion which is a linear term typable in the Curry type system (which is the simplest type system for the λ-calculus).
- Linear versions are preserved by weak head reduction, a notion of reduction considered by functional language compilers. This shows that expansion preserves computational behaviors of programs.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

These results have theoretical implications because we fully characterize a large class of functions (the functions definable by the strongly normalizable terms) with a rather simple class: the linear λ-terms. They also have practical implications because, although our notion of *expansion* is not decidable (because otherwise knowing if a term is strongly normalizable would be decidable, which it is not) it characterizes the same set of terms as intersection type systems for which there are decidable classes with impact in programming language technology (van Bakel, 1993; Jim, 1996; Kfoury & Wells, 1999). This suggests that decidable restrictions of our framework may be successfully defined with applications in program transformation.

In this paper we assume that the reader is familiar with the λ-calculus and type systems for the λ-calculus. Good surveys of the area are Barendregt (1992) and Hindley (1997).

We start in section 2 defining the type systems that are going to be used later in the paper. Section 3 defines the notion of expansion. In section 4 we show that expansion characterizes strong normalization. In section 5 we show that linear versions preserve types under a standard type transformation. In section 6 we show that expansion is preserved by several notions of reduction. Finally, we present the related work and point some directions for future work.

2 Types

Here we describe the type systems used in the rest of the paper.

2.1 Curry Types

The Curry type system (Curry, 1934) (sometimes called the Simple type system) is the basis of every type system for the λ-calculus. Here we describe this system along the lines presented in Hindley (1997). From now on terms of the λ-calculus are considered module α-equivalence. We also assume that in a term M no variable is bound more than once and no variable occurs both free and bound in M.

Definition 1
An infinite sequence of type-variables is assumed to be given. *Simple types* are expressions defined thus:

1. each type-variable is a simple type;
2. if σ and τ are simple types then $(\tau \to \sigma)$ is a simple type.

Definition 2
A finite set of pairs of the form $x : \tau$, where x is a term variable and τ is a simple type, is *consistent* if and only if the term variables are all distinct.

Definition 3
A *basis* is a consistent finite set of pairs of the form $x : \tau$, where x is a term variable and τ is a simple type.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Simple type derivations are defined thus:

$$VAR \qquad \{x : \tau\} \vdash_C x : \tau$$

$$ABS - I \qquad \frac{\Gamma \cup \{x : \tau\} \vdash_C M : \sigma}{\Gamma \vdash_C \lambda x.M : \tau \to \sigma}, \text{ if } \Gamma \cup \{x : \tau\} \text{ is consistent}$$

$$ABS - K \qquad \frac{\Gamma \vdash_C M : \sigma}{\Gamma \vdash_C \lambda x.M : \tau \to \sigma}, \text{ if } x \notin FV(M)$$

$$APP \qquad \frac{\Gamma_1 \vdash_C M : \tau \to \sigma, \Gamma_2 \vdash_C N : \tau}{\Gamma_1 \cup \Gamma_2 \vdash_C MN : \sigma}, \text{ if } \Gamma_1 \cup \Gamma_2 \text{ is consistent}$$

$M : \sigma$ is derivable from a basis Γ in the Curry type system, notation $\Gamma \vdash_C M : \sigma$, if and only if it is obtained using the previous rules.

Usually the Curry type system is defined using the same basis in the application rule and a variable rule of the form:

$$\Gamma \vdash x : \tau, \text{ if } x : \tau \in \Gamma$$

The definition presented here guarantees that every type in the basis is used in the type derivation. This feature is going to be important in subsequent results.

Notice that the two definitions are equivalent in the sense that $\Gamma_1 \vdash_C M : \sigma$ if and only if there is a basis $\Gamma_2 \supseteq \Gamma_1$ such that $\Gamma_2 \vdash M : \sigma$ in the usual definition.

2.2 Intersection types

Intersection types were introduced in Coppo & Dezani-Ciancaglini (1980) as powerful type systems with the ability of typing different occurrences of the same variable with different types. This makes it possible to type 'dangerous' self-applications such as $\lambda x.xx$. Notice that $\lambda x.xx$ is not typable in the Curry type system and in ML. Here we define an intersection type system where every type declared in the environment is used in the type derivation, a property which is going to be crucial in subsequent results.

Definition 4

An infinite sequence of type-variables is assumed to be given. *Intersection types* are expressions defined thus:

1. each type-variable is a type;
2. if σ and $\tau_1 \ldots \tau_n$ are types (for $n \geqslant 1$) then $(\tau_1 \cap \ldots \cap \tau_n \to \sigma)$ is a type.

We treat $\cap$ in an associative and commutative manner. Note that in contrast with the Coppo–Dezani type system $\cap$ is not idempotent. Thus $\alpha \cap \alpha \to \beta$ is not the same as $\alpha \cap \alpha \cap \alpha \to \beta$. This property guarantees that if a variable x occurs free in a term M then the types for x in the linearization of $\lambda x.M$ are in a one-to-one relation with the number of occurrences of x in M.

Definition 5

A *type environment* is a finite set of pairs of the form $x : \tau_1 \cap \ldots \cap \tau_n$, where x is a term variable, $\tau_1 \ldots \tau_n$ are types, and the term variables are all distinct.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Definition 6
Let Γ_1 and Γ_2 be two type environments. Then $\Gamma_1 \wedge \Gamma_2$ is the new environment given by $x : \sigma \in \Gamma_1 \wedge \Gamma_2$ if and only if σ is defined thus

$$\sigma = \begin{cases} \sigma_1 \cap \sigma_2 & \text{if } x : \sigma_1 \in \Gamma_1 \text{ and } x : \sigma_2 \in \Gamma_2 \\ \sigma_1 & \text{if } x : \sigma_1 \in \Gamma_1 \text{ and } \neg\exists\sigma.x : \sigma \in \Gamma_2 \\ \sigma_2 & \text{if } x : \sigma_2 \in \Gamma_2 \text{ and } \neg\exists\sigma.x : \sigma \in \Gamma_1 \end{cases}$$

The intersection type system used in the rest of the paper is defined thus:

$$VAR \qquad \{x : \tau\} \vdash x : \tau$$

$$ABS - I \qquad \frac{\Gamma \cup \{x : \tau_1 \cap \ldots \cap \tau_n\} \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau_1 \cap \ldots \cap \tau_n \to \sigma}$$

$$ABS - K \qquad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \to \sigma}, \text{ if } x \notin FV(M)$$

$$APP \qquad \frac{\Gamma_0 \vdash M : \tau_1 \cap \ldots \cap \tau_n \to \sigma, \Gamma_1 \vdash N : \tau_1 \ldots \Gamma_n \vdash N : \tau_n}{\Gamma_0 \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_n \vdash MN : \sigma}$$

The two different ABS rules are necessary because in this system if there is a derivation of $\Gamma \vdash M : \sigma$ and $x \notin FV(M)$ then there is not a type declaration for x in Γ. The set of types for a given term M in this system is strictly included in the set of types for M in the original intersection type system of Coppo & Dezani-Ciancaglini (1980). For example the type $(\alpha_1 \cap \alpha_2) \to \alpha_1$ types $\lambda x.x$ in the Coppo–Dezani type system, but not in the system used in this paper. The reason for this is that types in intersections for free variables can only be introduced with the APP rule and thus each element of the intersection corresponds to a type that is actually used in the type derivation. However, the set of terms typable in both systems is the same and corresponds to the strongly normalizable terms.

Theorem 1
A λ-term M is strongly normalizable (i.e. with no infinite reduction sequences starting from M) if and only if M is typable in the intersection type system presented.

Proof (Sketch)
The *if* part is proved by transforming a derivation in our type system in a derivation in the Coppo–Dezani type system (which types all strongly normalizable terms). This can be done by induction in the length of the derivation tree. The *only-if* part is similar to the proof of the same property for the Coppo–Dezani type system presented in Amadio & Curien (1998). The technique used is to show that if a term $M[N/x]$ is typable in an intersection type system with type τ then the redex $(\lambda x.M)N$ is also typable with the same type. The result follows by lifting this property to arbitrary terms using induction on the size of the term and on the maximal length of derivations starting in the term. □

One peculiarity of this type system is that it does not satisfy the property of *subject reduction* as it is shown by the following example:

Example 1
In this system

$$\{z : \alpha_2 \to \beta\} \vdash \lambda x.(\lambda y.z)xx : \alpha_1 \cap \alpha_2 \to \beta$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

and

$$\lambda x.(\lambda y.z)xx \underset{\beta}{\rightarrow} \lambda x.zx$$

but

$$\{z : \alpha_2 \rightarrow \beta\} \not\vdash \lambda x.zx : \alpha_1 \cap \alpha_2 \rightarrow \beta$$

The lack of subject reduction also happens in other restrictions of intersection type systems which are *relevant* in the sense that every type in the environment has to be used in the type derivation (Coppo & Giannini, 1995; Damiani & Giannini, 1994; Kfoury, 1996; Kfoury & Wells, 1999). This property will be crucial in our work since type information will direct the transformation from terms into linear terms.

3 Expansions

In this section we define the notion of expansion of a λ-term. Expansion consists of replacing each occurrence of a variable in a term by a new variable. This operation may involve other transformations in the term. For example if x is expanded k times in $(\lambda x.M)N$ then N has to be copied k times. However if the expansion is inside N then M may be changed, because possible arguments of x may have to be copied.

To define the expansion we face one key problem: the expansion of MN is a term of the form $M_0N_1 \ldots N_k$ where M_0 is the expansion of M and $N_1 \ldots N_k$ are expansions of N. The problem here is to find the right k. It is easy to determinate the number of new arguments when M is of the form $\lambda x.M'$ (just check how many fresh variables replace x), but if M is itself an application this information depends on expansions made inside M. The best way to propagate this information is by using types. As we need to explicitly count the number of types of each function argument we use intersection types. If M has type $\tau_1 \cap \ldots \cap \tau_k \rightarrow \sigma$ in the intersection type system we know that MN will be expanded to a term of the form $M_0N_1 \ldots N_k$. We now formalize these ideas by defining the expansion operation.

Let us first formalize the expansion of free variables.

Definition 7
A *variable expansion* is an expression of the form

$$x : S$$

where x is a variable and S is a set of pairs of the form $y : \tau$ where y is a variable and τ an intersection type. ($x : S$ should be read informally as "x expands to the variables in S".)

Definition 8
An *expansion context* A is any finite set of variable expansions

$$A = \{x_1 : S_1, \ldots, x_n : S_n\}$$

where the variables $\{x_1 \ldots x_n\}$ are all different and the S_i are disjoint.

We now define an operation which appends two expansion contexts.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Definition 9

Let A_1 and A_2 be two expansion contexts. Then $A_1 \uplus A_2$ is a new context such that $x : S \in A_1 \uplus A_2$ if and only if

$$S = \begin{cases} S_1 \cup S_2 & \text{if } x : S_1 \in A_1 \text{ and } x : S_2 \in A_2 \\ S_1 & \text{if } x : S_1 \in A_1 \text{ and } \neg\exists S.x : S \in A_2 \\ S_2 & \text{if } x : S_2 \in A_2 \text{ and } \neg\exists S.x : S \in A_1 \end{cases}$$

From now on when we write $A \uplus \{x : S\}$ we assume that x does not occur in A. We are now able to formalize the notion of term expansion:

Definition 10

Given a pair $M : \sigma$, where M is a term and σ an intersection type, a term N and an expansion context A we define a relation $\mathscr{E}(M : \sigma) \lhd (N, A)$ called *expansion*. If A is empty we shall write just $\mathscr{E}(M : \sigma) \lhd N$. Expansions are defined by

$$\mathscr{E}(x : \tau) \lhd (y, \{x : \{y : \tau\}\})$$
if x is a variable and y is a fresh variable

$$\mathscr{E}(\lambda x.M : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (\lambda x_1 \ldots x_n.M^*, A)$$
if x occurs in M and
$\mathscr{E}(M : \sigma) \lhd (M^*, A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\})$

$$\mathscr{E}(\lambda x.M : \tau \to \sigma) \lhd (\lambda y.M^*, A)$$
if x does not occur in M,
y is a fresh variable and
$\mathscr{E}(M : \sigma) \lhd (M^*, A)$

$$\mathscr{E}(MN : \sigma) \lhd (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \ldots \uplus A_n)$$
if for some $k > 0$ and $\tau_1, \ldots \tau_k$,
$\mathscr{E}(M : \tau_1 \cap \ldots \cap \tau_k \to \sigma) \lhd (M_0, A_0)$ and
$\mathscr{E}(N : \tau_i) \lhd (N_i, A_i), (1 \leqslant i \leqslant k)$

Remark: it is possible that in order to define $\mathscr{E}(M : \sigma)$ we must take a representation of σ with some redundancy, e.g. instead of:

$$\mathscr{E}(\lambda x.x(xx) : ((\alpha \to \alpha) \cap \alpha) \to \alpha)$$

we may require:

$$\mathscr{E}(\lambda x.x(xx) : ((\alpha \to \alpha) \cap (\alpha \to \alpha) \cap \alpha) \to \alpha)$$

in order to get a number of types in the intersection which is the same as the free occurrences of the parameter in the function body.

As we shall see, expansion relates λ-terms with their linear versions. By linear we mean exactly the following set of terms:

Definition 11

A linear λ-term is a λ-term M such that:

1. for each subterm $\lambda x.N$ of M, x occurs free in N at most once,
2. each free variable of M has just one occurrence in M.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

This definition follows the definition of linear terms presented in Kfoury (2000). Some people call this class of terms *affine*, and use the word *linear* for terms $\lambda x.M$ where x occurs free exactly once in M.

Lemma 1
For all N, $M : \sigma$ and A, if $\mathscr{E}(M : \sigma) \lhd (N, A)$, then N is a linear term.

Proof
By structural induction on M. The cases where M is a variable and a term of the form $\lambda x.N$ are trivial by structural induction. We just present the case where $M \equiv M_1 M_2$. In this case

$$\mathscr{E}(M_1 M_2 : \sigma) \lhd (M_0 N_1 \ldots N_k, A_0 \uplus A_1 \uplus \ldots \uplus A_n)$$

and

1. $\mathscr{E}(M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (M_0, A_0)$
2. $\mathscr{E}(M_2 : \tau_i) \lhd (N_i, A_i), (1 \leqslant i \leqslant k)$

By the induction hypothesis M_0 and $N_1, \ldots, N_k$ are linear. Notice that the free variables of M_0, $N_1, \ldots, N_k$ are all different, because the expansion of an occurrence of a variable creates a fresh variable. Thus $M_0 N_1 \ldots N_k$ is linear. □

Lemma 2
Let $\mathscr{E}(M : \sigma) \lhd (N, A \uplus \{x : \{x_1 : \tau_1, \ldots, x_k : \tau_k\}\})$. Then x occurs free in M k times.

Proof
By structural induction on M.

1. Base case: trivial by the definition of expansion.
2. Induction step:
   (a) $M \equiv \lambda y.M_1$. In this case by the induction hypothesis x occurs free in M_1 k times. Thus the same happens with $\lambda y.M_1$.
   (b) $M \equiv M_1 M_2$. The result follows applying the induction hypothesis to M_1 and M_2 and noticing that the variables $\{x_1, \ldots, x_k\}$ are all distinct, because any occurrence of a variable in the expansion of a term has exactly one occurrence.

□

From now on if $\mathscr{E}(M : \sigma) \lhd (N, A)$ we will refer to N as one linear version of M. We now present some examples.

Example 2
Let $I \equiv \lambda x.x$ and $M \equiv \lambda x.xx$. Let us show step by step how to calculate an expansion of $(MI : \alpha \to \alpha)$:

$$\mathscr{E}(x : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd (x_1, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

and

$$\mathscr{E}(x : \alpha \to \alpha) \lhd (x_2, \{x : \{x_2 : \alpha \to \alpha\}\})$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

thus

$$\mathscr{E}(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

and

$$\mathscr{E}(\lambda x.xx : (((\alpha \to \alpha) \to (\alpha \to \alpha)) \cap (\alpha \to \alpha)) \to \alpha \to \alpha) \lhd \lambda x_1 x_2 . x_1 x_2$$

It easy to show that

$$\mathscr{E}(I : \alpha \to \alpha) \lhd I$$

and

$$\mathscr{E}(I : (\alpha \to \alpha) \to (\alpha \to \alpha)) \lhd I$$

thus

$$\mathscr{E}(((\lambda x.xx)I) : \alpha \to \alpha) \lhd (\lambda x_1 x_2 . x_1 x_2) I I$$

Note that if

$$\mathscr{E}(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}\})$$

it is also true that

$$\mathscr{E}(xx : \alpha \to \alpha) \lhd (x_1 x_2, \{x : \{x_2 : \alpha \to \alpha, x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha)\}\})$$

because $\{x_1 : (\alpha \to \alpha) \to (\alpha \to \alpha), x_2 : \alpha \to \alpha\}$ is a set and thus there is not a fixed order among its elements. Thus we also have

$$\mathscr{E}(\lambda x.xx : ((\alpha \to \alpha) \cap ((\alpha \to \alpha) \to (\alpha \to \alpha)))) \to \alpha \to \alpha) \lhd \lambda x_2 x_1 . x_1 x_2$$

and consequently

$$\mathscr{E}(((\lambda x.xx)I) : \alpha \to \alpha) \lhd (\lambda x_2 x_1 . x_1 x_2) I I$$

Example 3
Let $M \equiv (\lambda xy.xy)(\lambda x.xx)z$. Then

$$\mathscr{E}(M : \beta) \lhd ((\lambda x_1 y_1 y_2 . x_1 y_1 y_2)(\lambda x_2 x_3 . x_2 x_3) z_1 z_2, \{z : \{z_1 : \alpha \to \beta, z_2 : \alpha\}\})$$

Example 4
Let $I \equiv \lambda x.x$ and $M \equiv (\lambda f.f(\lambda x.xx)(fI))I$ Then

$$\mathscr{E}(M : \alpha \to \alpha) \lhd (\lambda f_1 f_2 f_3 . f_1 (\lambda x_1 x_2 . x_1 x_2)(f_2 I)(f_3 I)) I I I$$

Notice in this example the use of type information to control the number of expansions. In$(\lambda f.f(\lambda x.xx)(fI))I$ the fact that f is going to be the identity function gives f three different types, one for the identity function applied to $\lambda x.xx$, and two more types, one for each type in the intersection in the argument type of $f(\lambda x.xx)$. These three types give rise to the three new expansion variables f_1, f_2 and f_3. The intersection of two types in the argument type of $f(\lambda x.xx)$ gives rise to the two new terms $(f_2 I)$ and $(f_3 I)$.

Example 5
Let $M \equiv (\lambda vxyz.v(y(vxz)))I$. Then

$$\mathscr{E}(M : (\alpha \to \beta) \to (\beta \to \sigma) \to \alpha \to \sigma) \lhd (\lambda v_1 v_2 xyz . v_1 (y(v_2 xz))) I I$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

This last example is used in Hindley's book (Hindley, 1997) to show that a term M may have a principal type in the Curry type system which is less general than the principal type of its normal form. In fact, in the Curry type system, M has principal type $(\alpha \to \beta) \to (\beta \to \alpha \to \beta) \to \alpha \to \alpha \to \beta$ and its normal form, $\lambda xyz.y(xz)$, has principal type $(\alpha \to \beta) \to (\beta \to \sigma) \to \alpha \to \sigma$. The reason for this is that in M the two occurrences of v must have the same type, but when v is replaced by I during the reduction process we drop that restriction. The interesting fact is that the expansion of M, $\lambda v_1 v_2 xyz.v_1(y(v_2 xz)))II$, has type $(\alpha \to \beta) \to (\beta \to \sigma) \to \alpha \to \sigma$ in the Curry type system, because when we expand v we do not have to impose the same type for v_1 and v_2.

4 Strong normalization

Here we show that terms that we can expand are exactly the terms typable in an Intersection Type System, i.e. the strongly normalizable terms. This result shows that there is a clear connection between intersection type systems, which give different types to different occurrences of the same variable and expansion where each occurrence of a variable is replaced by a new variable.

Let us first define two functions which transform expansion contexts in type environments and vice versa.

Definition 12
Let Γ be a type environment and $\{x_1, \ldots x_n\}$ be fresh term variables. $e(\Gamma)$ is the expansion context defined thus:

$$e(\Gamma) = \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\} | x : \tau_1 \cap \ldots \cap \tau_n \in \Gamma\}$$

Definition 13
Let A be an expansion context. $l(A)$ is the type environment defined thus:

$$l(A) = \{x : \tau_1 \cap \ldots \cap \tau_n | x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\} \in A\}$$

Lemma 3
Let Γ_1 and Γ_2 be type environments. Then

$$e(\Gamma_1) \uplus e(\Gamma_2) = e(\Gamma_1 \wedge \Gamma_2)$$

Proof
By the definitions of $\wedge$ and $\uplus$. □

Lemma 4
Let A_1 and A_2 be two expansion contexts. Then

$$l(A_1) \wedge l(A_2) = l(A_1 \uplus A_2)$$

Proof
By the definitions of $\wedge$ and $\uplus$. □

Lemma 5
Let M be a λ-term such that there is an environment Γ and an intersection type σ such that $\Gamma \vdash M : \sigma$. Then there is a linear term N such that $\mathscr{E}(M : \sigma) \lhd (N, e(\Gamma))$.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Proof
By structural induction on M.

1. Base case: M is a term-variable x. $\{x : \tau\} \vdash x : \tau$. Then $\mathscr{E}(x : \tau) \lhd (y, \{x : \{y : \tau\}\})$ where y is a fresh variable. The result follows noticing that $\{x : \{y : \tau\}\} = e(\{x : \tau\})$.
2. Induction step:
   (a) M is of the form $\lambda x.N$ and x occurs in N. In this case $\Gamma \cup \{x : \tau_1 \cap \ldots \cap \tau_n\} \vdash N : \sigma$. By the induction hypothesis,

$$\mathscr{E}(N : \sigma) \lhd (N', e(\Gamma) \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\})$$

   Thus by the definition of expansion

$$\mathscr{E}(\lambda x.N : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (\lambda x_1 \ldots x_n.N', e(\Gamma))$$

   (b) M is of the form $\lambda x.N$ and x does not occur in N. In this case $\Gamma \vdash N : \sigma$. By the induction hypothesis,

$$\mathscr{E}(N : \sigma) \lhd (N', e(\Gamma))$$

   and by the definition of expansion

$$\mathscr{E}(\lambda x.N : \tau \to \sigma) \lhd (\lambda y.N', e(\Gamma))$$

   (c) M is of the form $M_1 M_2$. In this case we have

$$\Gamma_0 \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_n \vdash M_1 M_2 : \sigma$$

   Thus

   i $\Gamma_0 \vdash M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma$

   ii $\Gamma_i \vdash M_2 : \tau_i (1 \leqslant i \leqslant n)$

   By the induction hypothesis:

   i $\mathscr{E}(M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (M_0, e(\Gamma_0))$

   ii $\mathscr{E}(M_2 : \tau_i) \lhd (N_i, e(\Gamma_i)), (1 \leqslant i \leqslant n)$

   Thus, by the definition of expansion,

$$\mathscr{E}(M_1 M_2 : \sigma) \lhd (M_0 N_1 \ldots N_n, e(\Gamma_0) \uplus \ldots \uplus e(\Gamma_n))$$

   and finally by lemma 3

$$\mathscr{E}(M_1 M_2 : \sigma) \lhd (M_0 N_1 \ldots N_n, e(\Gamma_0 \wedge \ldots \wedge \Gamma_n))$$

□

Lemma 6
Let M be a λ-term such that there is an expansion context A, an intersection type σ, and a linear term N such that $\mathscr{E}(M : \sigma) \lhd (N, A)$. Then $l(A) \vdash M : \sigma$.

Proof
By structural induction on M.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

1. Base case: M is a term variable x. In this case $\mathscr{E}(x : \tau) \lhd (y, \{x : \{y : \tau\}\})$. We have $l(\{x : \{y : \tau\}\}) = \{x : \tau\}$. Finally $\{x : \tau\} \vdash x : \tau$.
2. Induction step:
   (a) M is of the form $\lambda x.N$, and x occurs in N.

$$\mathscr{E}(\lambda x.N : (\tau_1 \cap \ldots \cap \tau_n \to \sigma)) \lhd (\lambda x_1 \ldots x_n.N', A)$$

   where

$$\mathscr{E}(N : \sigma) \lhd (N', A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\})$$

   By the induction hypothesis

$$l(A) \cup \{x : \tau_1 \cap \ldots \cap \tau_n\} \vdash N : \sigma$$

   Thus

$$l(A) \vdash \lambda x.N : \tau_1 \cap \ldots \cap \tau_n \to \sigma$$

   (b) M is of the form $\lambda x.N$ and x does not occur in N.

$$\mathscr{E}(\lambda x.N : \tau \to \sigma) \lhd (\lambda y.N', A)$$

   where

$$\mathscr{E}(N : \sigma) \lhd (N', A)$$

   By the induction hypothesis

$$l(A) \vdash N : \sigma$$

   Thus

$$l(A) \vdash \lambda x.N : \tau \to \sigma$$

   (c) M is of the form $M_1 M_2$. In this case we have:

$$\mathscr{E}((M_1 M_2) : \sigma) \lhd (M_0 N_1 \ldots N_k, A_0 \uplus \ldots \uplus A_k)$$

   and

   i $\mathscr{E}(M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (M_0, A_0)$
   ii $\mathscr{E}(M_2 : \tau_i) \lhd (N_i, A_i)(1 \leqslant i \leqslant n)$

   By the induction hypothesis

   i $l(A_0) \vdash M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma$
   ii $l(A_i) \vdash M_2 : \tau_i (1 \leqslant i \leqslant n)$

   Thus

$$l(A_0) \wedge \ldots \wedge l(A_n) \vdash M_1 M_2 : \sigma$$

   Finally, by lemma 4,

$$l(A_0 \uplus \ldots \uplus A_n) \vdash M_1 M_2 : \sigma$$

□

Theorem 2

Let M be a λ-term. Then M is strongly normalizable if and only if there are a linear term N, an expansion context A and a type σ such that $\mathscr{E}(M : \sigma) \lhd (N, A)$.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Proof
By theorem 1 the stronlgy normalizable terms are the terms typable in the intersection type system presented. The result follows by Lemmas 5 and 6. □

5 Expansion and types

In Bucciarelli *et al.* (1999), a translation from intersection types to Curry types was given and used to show that derivations in an intersection type system with idempotent intersections can be transformed into terms typed in the Curry type system. Here we show that our definition of expansion preserves this translation. In fact, let $\mathscr{T}$ be the translation from intersection types to Curry types defined in Bucciarelli *et al.* (1999). Then, if M is typable in the intersection type system with type σ, and $\mathscr{E}(M : \sigma) \lhd (N, A)$ then N is typable in the Curry type system with type $\mathscr{T}(\sigma)$.

Definition 14
$\mathscr{T}$ is a translation from intersection types to Curry types defined by:

1. $\mathscr{T}(\alpha) = \alpha$, if α is a type variable;
2. $\mathscr{T}((\tau_1 \cap \ldots \cap \tau_n) \to \sigma) = \mathscr{T}(\tau_1) \to \ldots \to \mathscr{T}(\tau_n) \to \mathscr{T}(\sigma)$.

The previous definition can be extended to expansion contexts:

Definition 15
Let $\mathscr{T}_e$ be a translation from expansion contexts to bases defined thus:

1. $\mathscr{T}_e(\emptyset) = \emptyset$;
2. $\mathscr{T}_e(A \cup \{x : \{x_1 : \tau_1, \ldots, x_n : \tau_n\}\}) = \mathscr{T}_e(A) \cup \{x_1 : \mathscr{T}(\tau_1), \ldots, x_n : \mathscr{T}(\tau_n)\}$.

Theorem 3
Let $\mathscr{E}(M : \sigma) \lhd (N, A)$. Then $\mathscr{T}_e(A) \vdash_C N : \mathscr{T}(\sigma)$, where $\vdash_C$ stands for type derivation in the Curry type system.

Proof
By structural induction on M.

1. Base case. M is a term variable x. In this case $\mathscr{E}(x : \tau) \lhd (y, \{x : \{y : \tau\}\})$. $\mathscr{T}_e(\{x : \{y : \tau\}\}) = \{y : \mathscr{T}(\tau)\}$. The result follows by the VAR rule for the Curry type system.
2. Induction step:
   (a) M is of the form $\lambda x.N$. Suppose that x occurs free in M. Then

$$\mathscr{E}(\lambda x.N : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (\lambda x_1 \ldots x_n.N^*, A)$$

   and thus

$$\mathscr{E}(N : \sigma) \lhd (N^*, A \cup \{x : \{x_1 : \tau_1 \ldots x_n : \tau_n\}\})$$

   By the induction hypothesis and the definition of $\mathscr{T}_e$:

$$\mathscr{T}_e(A) \cup \{x_1 : \mathscr{T}(\tau_1), \ldots, x_n : \mathscr{T}(\tau_n)\} \vdash_C N^* : \mathscr{T}(\sigma)$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Thus, by successive applications of the ABS-I rule:

$$\mathscr{T}_e(A) \vdash_C \lambda x_1 \ldots x_n.N^* : \mathscr{T}(\tau_1) \to \ldots \to \mathscr{T}(\tau_n) \to \mathscr{T}(\sigma)$$

The result follows by the definition of $\mathscr{T}$. The case where x does not occur in N is similar, thus we will omit it.

(b) M is of the form $M_1 M_2$. In this case:

i $\mathscr{E}(M_1 : \tau_1 \cap \ldots \cap \tau_n \to \sigma) \lhd (M_0, A_0)$

ii $\mathscr{E}(M_2 : \tau_i) \lhd (N_i, A_i)(1 \leqslant i \leqslant n)$

By the induction hypothesis

i $\mathscr{T}_e(A_0) \vdash_C M_0 : \mathscr{T}(\tau_1) \to \ldots \to \mathscr{T}(\tau_n) \to \mathscr{T}(\sigma)$

ii $\mathscr{T}_e(A_i) \vdash_C N_i : \mathscr{T}(\tau_i)$, for $(1 \leqslant i \leqslant n)$.

Notice that the variables in $A_0, \ldots, A_n$ are all distinct, because expansion contexts are generated with expansions of occurrences of free variables, which always generate fresh variables. Thus the same happens for $\mathscr{T}_e(A_0), \ldots, \mathscr{T}_e(A_n)$. This guarantees that in $\mathscr{T}_e(A_0) \cup \ldots \cup \mathscr{T}_e(A_n)$ all variables are distinct. Thus

$$\mathscr{T}_e(A_0) \cup \ldots \cup \mathscr{T}_e(A_n) \vdash_C M_0 N_1 \ldots N_n : \mathscr{T}(\sigma)$$

□

Theorem 4

Let M be a λ-term such that $\Gamma \vdash M : \sigma$ in the intersection type system. Then there is a basis Γ_C and a linear term N such that $\Gamma_C \vdash_C N : \mathscr{T}(\sigma)$, where $\vdash_C$ stands for type derivation in the Curry type system.

Proof

By lemma 5, $\Gamma \vdash M : \sigma \Rightarrow \mathscr{E}(M : \sigma) \lhd (N, e(A))$. The result follows by theorem 3. □

This theorem has, as a corollary, that if a term M is typable in the intersection type system with a Curry type, then there is a linear term with the same type derivable in the Curry type system. Just notice that $\mathscr{T}(\sigma) = \sigma$ when σ is a Curry type.

6 Expansion and reductions

In this section we show that expansion is preserved by a notion of reduction which is used in the implementation of functional programming languages: weak head reduction. This guarantees that the weak head normal form of a term M has a linear version which is a weak head normal form of a linear version of M.

We then show that expansion is preserved by β-reduction for the λI-calculus, where in any term of the form $\lambda x.M$, x has to occur free in M.

We first present one lemma which is going to be used in the study of the preservation of expansion by reduction.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Lemma 7
Let $\mathscr{E}(M : \sigma) \lhd (M_0, A_0 \uplus \{x : \{x_1 : \tau_1, \ldots, x_k : \tau_k\}\})$ and $\mathscr{E}(N : \tau_i) \lhd (N_i, A_i)$ for $i \in \{1, \ldots, k\}$. Then

$$\mathscr{E}(M[N/x] : \sigma) \lhd (M_0[N_1/x_1, \ldots, N_k/x_k], A_0 \uplus \ldots \uplus A_k)$$

Proof
The proof will follow by structural induction on M. Notice that, by lemma 2, x occurs free in M.

1. Base case: $M \equiv x$. In this case:

   $$\mathscr{E}(x : \sigma) \lhd (y, \{x : \{y : \sigma\}\})$$

   thus

   $$\mathscr{E}(x[N/x] : \sigma) \lhd (y[N_1/y], A_1)$$

   where

   $$\mathscr{E}(N : \sigma) \lhd (N_1, A_1)$$

2. Induction step:

   (a) $M \equiv \lambda y.M_0$. Assume that y occurs free in M_0. The other case is simpler. In this case:

   $$\mathscr{E}(\lambda y.M_0 : \delta_1 \cap \ldots \cap \delta_n \to \sigma_1) \lhd (\lambda y_1 \ldots y_n.M_0^*, A_0 \uplus \{x : \{x_1 : \tau_1, \ldots x_k : \tau_k\}\})$$

   by the definition of expansion we have

   $$\mathscr{E}(M_0 : \sigma_1) \lhd (M_0^*, A_0 \uplus \{x : \{x_1 : \tau_1, \ldots x_k : \tau_k\}\} \uplus \{y : \{y_1 : \delta_1, \ldots y_n : \delta_n\}\}$$

   and

   $$\mathscr{E}(N : \tau_i) \lhd (N_i, A_i), i \in \{1 \ldots k\}$$

   By the induction hypothesis it follows:

   $$\mathscr{E}(M_0[N/x] : \sigma_1) \lhd (M_0^*[N_1/x_1, \ldots, N_k/x_k], A_0 \uplus \ldots \uplus A_k \uplus \{y : \{y_1 : \delta_1, \ldots y_n : \delta_n\}\}$$

   thus

   $$\mathscr{E}(\lambda y.M_0)[N/x] : \delta_1 \cap \ldots \cap \delta_n \to \sigma_1) \lhd (\lambda y_1 \ldots y_n.M_0^*[N_1/x_1, \ldots, N_k/x_k], A_0 \uplus \ldots \uplus A_k)$$

   (b) $M \equiv M_1 M_2$.

   $$\mathscr{E}(M_1 M_2 : \sigma) \lhd (P_0 P_1 \ldots P_n, B_0 \uplus B_1 \uplus \ldots \uplus B_n \uplus \{x : \{x_1 : \tau_1, \ldots, x_k : \tau_k\}\})$$

   and

   $$\mathscr{E}(N : \tau_i) \lhd (N_i, A_i), i \in \{1, \ldots, k\}$$

   Let $X = \{x : \{x_1 : \tau_1, \ldots, x_k : \tau_k\}\} = X_0 \uplus \ldots \uplus X_n$ where $X_i = \{x : \{x_1^i : \tau_1^i, \ldots, x_{k_i}^i : \tau_{k_i}^i\}\}$ and $\{x_1^i, \ldots, x_{k_i}^i\}$ is the subset of $\{x_1, \ldots, x_k\}$ whose elements occur in P_i for $i \in \{0, \ldots, n\}$. Now we have

   $$\mathscr{E}(M_1 : \delta_1 \cap \ldots \cap \delta_n \to \sigma) \lhd (P_0, B_0 \uplus X_0)$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

and

$$\mathscr{E}(M_2 : \delta_i) \lhd (P_i, B_i \uplus X_i), i \in \{1, \ldots, n\}$$

Let $T = \{N_1, \ldots, N_k\} = T_0 \cup \ldots \cup T_n$ where $T_i = \{N_1^i, \ldots, N_{k_i}^i\}$ is the subset of T whose elements occur in $P_i[N_1/x_1, \ldots, N_k/x_k]$ and such that $\mathscr{E}(N : \tau_j^i) \lhd (N_j^i, A_j)$ for $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k_i\}$. By the induction hypothesis we have:

$$\mathscr{E}(M_1[N/x] : \delta_1 \cap \ldots \cap \delta_n \to \delta) \lhd \left(P_0\left[x_1^0/N_1^0, \ldots, x_{k_0}^0/N_{k_0}^0\right], B_0\right)$$

and

$$\mathscr{E}(M_2[N/x] : \delta_i) \lhd \left(P_i\left[x_1^i/N_1^i, \ldots, x_{k_i}^i/N_{k_i}^i\right], B_i\right), i \in \{1, \ldots, n\})$$

Thus

$$\mathscr{E}((M_1 M_2)[N/x] : \sigma) \lhd ((P_0 P_1 \ldots P_n)[x_1/N_1, \ldots, x_k/N_k], B_0 \uplus B_1 \uplus \ldots \uplus B_n)$$

□

6.1 Weak head reduction

Functional language compilers consider only weak-head reduction and stop evaluation when a weak-head normal form (a constant or a λ-abstraction) is reached. Weak-head normal forms are sufficient because printable results only belong to basic domains. Details about this subject can be found in Peyton Jones (1987). The following definition of *weak head reduction* appears in Fradet (1994):

Definition 16
Weak head reduction $\underset{w}{\to}$ is defined by:

$$(\lambda x.M)N \underset{w}{\to} M[N/x]$$

and

$$\frac{M \underset{w}{\to} M'}{MN \underset{w}{\to} M'N}$$

We denote by $\underset{w}{\twoheadrightarrow}$ the reflexive and transitive closure of $\underset{w}{\to}$. Closed weak head normal forms are abstractions $\lambda x.M$.

We first define an inclusion relation between expansion contexts as follows:

Definition 17
Let A_1 and A_2 be two expansion contexts. $A_1 \sqsubseteq A_2$ if and only if:

$$x : S_1 \in A_1 \Rightarrow x : S_2 \in A_2 \text{ and } S_1 \subseteq S_2$$

Lemma 8
Let $(\lambda x.M)N$ be a redex in the λ-calculus. Let

$$\mathscr{E}((\lambda x.M)N : \sigma) \lhd (N_1, A_1)$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Then there is a term N_2 such that

$$\mathscr{E}(M[N/x] : \sigma) \lhd (N_2, A_2)$$

,$A_2 \sqsubseteq A_1$ and $N_1 \underset{\beta}{\twoheadrightarrow} N_2$.

Proof
We consider two cases:

1. $x \in FV(M)$. By the definition of expansion:

   $$\mathscr{E}((\lambda x.M)N : \sigma) \lhd ((\lambda x_1 \ldots x_k.M_0)N_1 \ldots N_k, A_0 \uplus \ldots \uplus A_k)$$

   where

   $$\mathscr{E}((\lambda x.M) : \tau_1 \cap \ldots \cap \tau_k \to \sigma) \lhd (\lambda x_1 \ldots x_k.M_0, A_0)$$

   and

   $$\mathscr{E}(N : \tau_i) \lhd (N_i, A_i)$$

   Then we have

   $$\mathscr{E}(M : \sigma) \lhd (M_0, A_0 \uplus \{x : \{x_1 : \tau_1 \ldots x_k : \tau_k\}\})$$

   By lemma 7 we have:

   $$\mathscr{E}(M[N/x] : \sigma) \lhd (M_0[N_1/x_1, \ldots, N_k/x_k], A_0 \uplus \ldots \uplus A_k)$$

2. $x \notin FV(M)$. In this case

   $$\mathscr{E}((\lambda x.M)N : \sigma) \lhd ((\lambda y.M_0)N_0, A_0 \uplus A_1)$$

   where

   $$\mathscr{E}(\lambda x.M : \tau \to \sigma) \lhd (\lambda y.M_0, A_0)$$

   and

   $$\mathscr{E}(N : \tau) \lhd (N_0, A_1)$$

   Thus

   $$\mathscr{E}(M : \sigma) \lhd (M_0, A_0)$$

   and

   $$\mathscr{E}(M[N/x] : \sigma) = \mathscr{E}(M : \sigma) \lhd (M_0, A_0)$$

   Note that $A_0 \sqsubseteq A_0 \uplus A_1$.

□

To show that expansion is preserved by weak head reduction we need the concept of *context* as a term containing one hole [ ].

Definition 18
Contexts $C[\ ]$ are described by

1. [ ] is a context;
2. If $C[\ ]$ is a context and M a λ-term, then $C[\ ]M$, $MC[\ ]$ and $\lambda x.C[\ ]$ are contexts.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

If M is a λ-term and $C[\,]$ a context then $C[M]$ is the result of replacing the hole in $C[\,]$ with M. Note that this operation is different from that of substitution because no renaming of bound variables is allowed.

Theorem 5
Let $\mathscr{E}(M_1 : \sigma) \lhd (N_1, A_1)$ and $M_1 \underset{w}{\rightarrow} M_2$. Then there is a term N_2 such that $\mathscr{E}(M_2 : \sigma) \lhd (N_2, A_2)$, $N_1 \underset{w}{\twoheadrightarrow} N_2$ and $A_2 \sqsubseteq A_1$.

Proof
We use structural induction on the context C such that $M_1 \overset{R}{\underset{w}{\rightarrow}} M_2$ and $M_1 \equiv C[R]$. By $M_1 \overset{R}{\underset{w}{\rightarrow}} M_2$ we mean that M_1 reduces to M_2 by reducing the redex R.

1. Base case: M_1 is the w-redex R. The proof follows from Lemma 8.
2. Induction step:
   (a) $M_1 \equiv C[R]W$. In this case

$$\mathscr{E}(C[R]W : \sigma) \lhd (P_0 P_1 \ldots P_k, A_0 \uplus A_1 \uplus \ldots \uplus A_k)$$

Thus

$$\mathscr{E}(C[R] : \tau_1 \cap \ldots \cap \tau_n \rightarrow \sigma) \lhd (P_0, A_0)$$

and

$$\mathscr{E}(W : \tau_i) \lhd (P_i, A_i)$$

By the induction hypothesis there is a term P_0^* such that

$$C[R] \underset{w}{\rightarrow} P$$

$$\mathscr{E}(P : \tau_1 \cap \ldots \cap \tau_n \rightarrow \sigma) \lhd (P_0^*, B_0)$$

$$B_0 \sqsubseteq A_0$$

and

$$P_0 \underset{w}{\twoheadrightarrow} P_0^*$$

Thus

$$\mathscr{E}(P W : \sigma) \lhd (P_0^* P_1 \ldots P_k, B_0 \uplus A_1 \uplus \ldots \uplus A_k)$$

and

$$P_0 P_1 \ldots P_k \underset{w}{\twoheadrightarrow} P_0^* P_1 \ldots P_k$$

Note that $B_0 \sqsubseteq A_0$, thus $B_0 \uplus A_1 \uplus \ldots \uplus A_k \sqsubseteq A_0 \uplus A_1 \uplus \ldots \uplus A_k$

□

Definition 19
Let t and u be w-reductions starting, respectively, by M_0 and N_0:

$$t : M_0 \underset{w}{\rightarrow} M_1 \underset{w}{\rightarrow} M_2 \underset{w}{\rightarrow} \cdots$$

$$u : N_0 \underset{w}{\twoheadrightarrow} N_1 \underset{w}{\twoheadrightarrow} N_2 \underset{w}{\twoheadrightarrow} \cdots$$

We say that u is an *expansion* of t if there are expansion contexts $A_0, \ldots, A_k$ and a

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

type σ such that:

1. $A_0 \sqsupseteq A_1 \sqsupseteq A_2 \cdots$,
2. $\mathscr{E}(M_i : \sigma) \lhd (N_i, A_i)$ for $i \geqslant 0$.

The following corollary of Theorem 5 makes explicit the simple fact that every finite w-reduction can be expanded.

Corollary 1 (*of Theorem 5*)
Every finite w-reduction t, can be expanded to another w-reduction (not necessarily unique).

Proof
Apply successively Theorem 5 to every w-reduction step in t. □

We saw that expansion is preserved by weak head reduction. This does not happen with β-reduction. In fact we may have $M_1 \underset{\beta}{\rightarrow} M_2$, $\mathscr{E}(M_1 : \sigma) \lhd (N_1, A_1)$ and there is not a type τ such that $\mathscr{E}(M_2 : \tau) \lhd (N_2, A_2)$ and $N_1 \underset{\beta}{\twoheadrightarrow} N_2$. Note that there is a linear version, P, of M_2 (because M_1 is strongly normalizable thus M_2 is also strongly normalizable, and thus, by Theorem 2, it has a linear version). The point here is that $N_1 \underset{\beta}{\not\twoheadrightarrow} P$ for no linear version P of M_2. To see this let $M_1 \equiv \lambda x.(\lambda y.z)xx$ and $M_2 \equiv \lambda x.zx$. We have:

$$\lambda x.(\lambda y.z)xx \underset{\beta}{\rightarrow} \lambda x.zx$$

$$\mathscr{E}(\lambda x.(\lambda y.z)xx : \alpha_1 \cap \alpha_2 \rightarrow \beta) \lhd (\lambda x_1 x_2.(\lambda y_1.z_1)x_1 x_2, \{z : \{z_1 : \alpha_2 \rightarrow \beta\}\})$$

and

$$\lambda x_1 x_2.(\lambda y_1.z_1)x_1 x_2 \underset{\beta}{\rightarrow} \lambda x_1 x_2.z_1 x_2$$

Now note that, as x occurs in zx once, it follows from Lemma 2 that any expansion of $\lambda x.zx$ is of the form $\lambda x_1.M$ where M is one expansion of zx. Thus $\lambda x_1 x_2.z_1 x_2$ cannot be an expansion of $\lambda x.zx$ for any type. In order to have preservation under β-reduction $\lambda x_1 x_2.z_1 x_2$ would have to be a linear version of $\lambda x.zx$, which goes against the initial intuition that if x occurs k times in M then the linear versions of $\lambda x.M$ are of the form $\lambda x_1 \ldots x_k.M'$ where each x_i replaces one occurrence of x in M.

This property is related to the lack of subject reduction of the intersection type system used to direct the expansion process. If preservation of expansion by β-reduction is not viewed as a goal by itself, then the lack of this property is not a problem, because it holds for a notion of reduction which is used for functional programming languages. Thus expansion preserves computational behavior of programs, and therefore can be used as the basis of a program transformation technique for linearization. Expansion also preserves types under the standard type transformation $\mathscr{T}$ and thus can be useful for type inference. The feature which prevent preservation of expansion by β-reduction makes the definition of linear versions much easier and intuitive.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

6.2 Expansions and the λI-calculus

The λI-calculus is a restriction of the λ-calculus where in terms of the form $\lambda x.M$, x occurs free in M. The lack of subject reduction happens in terms of the form $\lambda x.M$ when x may be erased in the reduction of M. Here we show that β-reduction is preserved for the λI-calculus, where erasing is not allowed.

Lemma 9
Let $(\lambda x.M)N$ be a redex in the λI-calculus. Let

$$\mathscr{E}((\lambda x.M)N : \sigma) \lhd (N_1, A)$$

Then there is a term N_2 such that

$$\mathscr{E}(M[N/x] : \sigma) \lhd (N_2, A)$$

and $N_1 \underset{\beta}{\twoheadrightarrow} N_2$.

Proof
The proof is identical to the case in the proof of Lemma 8 where x occurs free in M. □

Theorem 6
Let M_1 and M_2 be two terms in the λI-calculus. Let $\mathscr{E}(M_1 : \sigma) \lhd (N_1, A)$ and $M_1 \underset{\beta}{\rightarrow} M_2$. Then there is a term N_2 such that $\mathscr{E}(M_2 : \sigma) \lhd (N_2, A)$ and $N_1 \underset{\beta}{\twoheadrightarrow} N_2$.

Proof
We use structural induction on the context C such that $M_1 \overset{R}{\underset{\beta}{\rightarrow}} M_2$ and $M_1 \equiv C[R]$.

1. Base case: M_1 is the β-redex R. The proof follows from Lemma 9.
2. Induction step:
   (a) $M_1 = \lambda x.C[R]$. In this case:

$$\mathscr{E}(\lambda x.C[R] : \delta_1 \cap \ldots \cap \delta_k \rightarrow) \lhd (\lambda x_1 \ldots x_k.M^*, A)$$

x occurs free in $\lambda x.C[R]$, thus

$$\mathscr{E}(C[R] : \delta) \lhd (M^*, A \uplus \{x : \{x_1 : \delta_1, \ldots, x_k : \delta_k\}\})$$

By the induction hypothesis there is a term N_2 such that

$$C[R] \underset{\beta}{\rightarrow} P$$

and

$$\mathscr{E}(P : \delta) \lhd (N_2, A \uplus \{x : \{x_1 : \delta_1, \ldots, x_k : \delta_k\}\}$$

and

$$M^* \underset{\beta}{\twoheadrightarrow} N_2$$

Thus

$$\lambda x.C[R] \underset{\beta}{\rightarrow} \lambda x.P$$

$$\mathscr{E}(\lambda x.P : \delta_1 \cap \ldots \cap \delta_k \rightarrow \delta) \lhd (\lambda x_1 \ldots x_k.N_2, A)$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

and

$$\lambda x_1 \dots x_k.M^* \underset{\beta}{\twoheadrightarrow} \lambda x_1 \dots x_k.N_2$$

(b) $M_1 \equiv C[R]W$. In this case

$$\mathscr{E}(C[R]W : \sigma) \lhd (P_0 P_1 \dots P_k, A_0 \uplus A_1 \uplus \dots \uplus A_k)$$

Thus

$$\mathscr{E}(C[R] : \tau_1 \cap \dots \cap \tau_n \to \sigma) \lhd (P_0, A_0)$$

and

$$\mathscr{E}(W : \tau_i) \lhd (P_i, A_i)$$

By the induction hypothesis there is a term P_0^* such that

$$C[R] \underset{\beta}{\to} P$$
$$\mathscr{E}(P : \tau_1 \cap \dots \cap \tau_n \to \sigma) \lhd (P_0^*, A_0)$$

and

$$P_0 \underset{\beta}{\twoheadrightarrow} P_0^*$$

Thus

$$\mathscr{E}(P W : \sigma) \lhd (P_0^* P_1 \dots P_k, A_0 \uplus \dots \uplus A_k)$$

and

$$P_0 P_1 \dots P_k \underset{\beta}{\twoheadrightarrow} P_0^* P_1 \dots P_k$$

(c) Suppose that $M_1 C[R] \underset{\beta}{\to} M_1 N_2$. Thus

$$\mathscr{E}(M_1 C[R]) \lhd (P_0 P_1 \dots P_k, A_0 \uplus \dots \uplus A_k)$$

Thus

$$\mathscr{E}(M_1 : \tau_1 \cap \dots \cap \tau_k \to \sigma) \lhd (P_0, A_0)$$

and

$$\mathscr{E}(C[R] : \tau_i) \lhd (P_i, A_i), i \in \{1, \dots, k\}$$

By the induction hypothesis for $i \in \{1, \dots, k\}$ there are terms P_i^* such that

$$\mathscr{E}(N_2 : \tau_i) \lhd (P_i^*, A_i)$$

and

$$P_i \underset{\beta}{\twoheadrightarrow} P_i^*$$

Thus

$$\mathscr{E}(M_1 N_2) \lhd (P_0 P_1^* \dots P_k^*, A_0 \uplus \dots \uplus A_k)$$

and

$$P_0 P_1 \dots P_k \underset{\beta}{\twoheadrightarrow} P_0 P_1^* \dots P_k^*$$

□

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

7 Related work

7.1 Other approaches to linearization

The linearization problem was the topic of one previous paper by Kfoury (2000), where a new calculus was defined satisfying the linearity condition. This condition states that in an abstraction $\lambda x.M$ the free occurrences of x in M are in one-one correspondence with the arguments to which the function is applied. Kfoury enlarged the λ-calculus with a new kind of terms of the form $M.P_1 \wedge \ldots \wedge P_n$ (this new calculus was denoted $\Lambda^{\wedge}$) and a new reduction $\beta^{\wedge}$, which, when the redex is of the form $(\lambda x.M)P_1 \wedge \ldots \wedge P_n$ and x occurs free is M n times, replaces the i^{th} occurrence of x by P_i $(1 \leqslant i \leqslant n)$. This reduction was made in parallel for every P_i. Finally an intersection type system for this new calculus was defined and used to give another proof of the well-known equivalence between strongly normalizing λ-terms and terms typable in an intersection type system. The main differences in our approach are:

1. Kfoury forced the linearity condition by changing the calculus and the reduction rule. Our translation is inside the λ-calculus from non-linear terms into linear ones and we continue to use β-reduction. Note that requiring the linearity condition is weaker than requiring that a term is linear (the linearity condition holds for point 1 of Definition 11).
2. Kfoury needs an intersection type system to type the expanded versions of terms. In our framework expanded versions of terms belong to the linear λ-calculus which is typable in the Curry type system.
3. We use type information to handle linearization. By choosing the right type system the linearization process defined here becomes rather simple when compared to the linearization presented in Kfoury (2000).
4. In Kfoury (2000), intersection was not commutative. As we saw before, this means that some valid linear versions would be lost.

This last item, concerning non-commutative intersections, deserves further explanation. Note that in Kfoury (2000), reduction was order-sensitive, in the sense that when $(\lambda x.M).P_1 \wedge \ldots \wedge P_n$ reduces to $M[P_1/x^1, \ldots, P_n/x^n]$, $x^1, \ldots, x^n$ are the occurrences of x in M numbered when M is scanned from left to right. Thus, a notion of expansion similar to ours corresponding to Kfoury's calculus had to use non-commutative intersections. For example the expansion of $M \equiv (\lambda x.xx)k$ would be $(\lambda x_1 x_2.x_1 x_2)k_1 k_2$ and, without commutative intersections, we would not be able to relate M with $(\lambda x_2 x_1.x_1 x_2)k_2 k_1$ which is also a valid linear version of M.

The lack of commutativity of the intersection operator has more serious implications concerning preservation of computational properties after expansion. Consider the following example:

Example 6
Let $M \equiv (\lambda x.(\lambda y.yx)x)z$. M has weak head normal form zz, which expands to $z_1 z_2$. The expansion of M would be the linear term $N \equiv (\lambda x_1 x_2.(\lambda y.yx_1)x_2)z_1 z_2$. Note that N has weak head normal form $z_2 z_1$. Thus, without commutative intersections, we loose preservation of weak head normal forms by expansion.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

Preservation of expansion by weak head reductions would still be possible for non-commutative intersections if we considered terms modulo permutation of free variables and expansion contexts modulo permutation of variables resulting from expansion. However, this would unnecessarily complicate the proofs without gaining anything with it (in fact we would loose valid linear versions as shown before).

A deeper consequence of the lack of commutative intersections is the lost of preservation of expansion by β-reduction for the λI-calculus. Consider the following example:

Example 7
In the λI-calculus, let $M \equiv \lambda x.(\lambda y.yx)x$. Without commutative intersections, M expands to the term $N \equiv \lambda x_1 x_2.(\lambda y.yx_1)x_2$. N has β-normal form $\lambda x_1 x_2.x_2 x_1$ which, without commutative intersections, is not the expansion of the normal form of M, $\lambda x.xx$.

These examples show that Kfoury's calculus relates to a notion of linearization inside the λ-calculus which does not correspond to our definition. It will be a useful investigation (left to others) whether a suitable characterization of expanded terms corresponding to Kfoury's calculus (for example, based on linear terms modulo certain properties) would keep the nice computational properties presented here.

7.2 Intersection types

A translation from derivations in a standard intersection type system (with associative, commutative and idempotent intersections) to terms typable in the Curry type system was presented in Bucciarelli *et al.* (1999). The authors used this result to prove that all functions uniformly defined using intersection types are also definable using Curry types. Here we characterize terms typable in an intersection type system with a more restrictive class of terms: the linear terms. In order to get linearization, one cannot have idempotent intersections and every assumption must be used in the type derivation process (otherwise, types would not give us information about the number of variables which occur in a term). This difference has a drastic effect in the computational behavior of expansion which is the lack of preservation of expansion by β-reduction. We show that a correct definition of expansion is still possible, when considered with respect to weak head reduction. Another difference from the translation in Bucciarelli *et al.* (1999) is that our definition of expansion uses induction on the terms, which quite simplifies the proofs.

Our definition of expansion is based on a type system where every assumption is used in the type derivation process. Other intersection type systems with that property were defined before. Particularly noteworthy are the systems defined in Kfoury & Wells (1999) and Kfoury (1996). The main difference to our system is that in Kfoury & Wells (1999) and Kfoury (1996), intersection is not commutative. Concerning linearization, with a non-commutative $\cap$ we would loose some valid linear versions of terms. As an example, we would not have $(\lambda x_2 x_1.x_1 x_2)II$ as a linear version of $(\lambda x.xx)I$ (see example 2). Other type systems with the same property are presented in Coppo & Giannini (1995) and Damiani & Giannini (1994). The

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

motivation in these systems was the definition of *relevant* decidable fragments of intersection type systems where the definition of relevance is stronger than only using all assumptions in derivations. This led to more complex systems than we need for our purposes.

Finally, the linear λ-calculus was also studied in a different context by the linear logic community (Abramsky, 1993; Lincoln & Mitchell, 1992; Wadler, 1990; Benton *et al*., 1992). In these works, the linear λ-calculus is defined as the computational interpretation of linear logic and the problem of transforming non-linear terms into linear ones is not a central issue.

8 Conclusion

We have introduced the notion of *expansion*, developed its basic properties, established its relationship with intersection type systems and used it to get a simple and direct definition of linearization of the λ-calculus.

Despite not being decidable, expansion can be used as the basis of new program transformation techniques for linearization. Our work suggests that these transformations should be related to decidable restrictions of intersection types.

Acknowledgements

We are grateful to Sabine Broda, Nelma Moreira, Maribel Fernandez and Ian Mackie for discussions on the subject of linearization. We also would like to thank the anonymous referees for their comments and suggestions.

This work partially supported by funds granted to *LIACC* through the *Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia* and *Programa POSI*.

A Strong normalization for intersection types

Here we present a proof of Theorem 1 and several definitions and lemmas needed to the main proof.

Definition 20
The size of a term M is defined as:

1. $size(x) = 1$
2. $size(MN) = size(M) + size(N) + 1$
3. $size(\lambda x.M) = size(M) + 1$

If M is strongly normalizable, the maximal length of a derivation starting from M is called the reduction depth of M and is denoted $depth(M)$.

Definition 21
Given two type environments Γ_1 and Γ_2, $\Gamma_1 \subseteq \Gamma_2$ if and only if $\exists_{\Gamma_3}.\Gamma_2 = \Gamma_1 \wedge \Gamma_3$.

Definition 22
The size of a type σ, where α ranges over the set of type variables, is defined by:

1. $size(\alpha) = 1$
2. $size(\sigma \to \tau) = size(\sigma) + size(\tau) + 1$

Definition 23
Given a type environment Γ and a variable x, the set of declared types for x in Γ, denoted by $types(x, \Gamma)$, is defined by:

1. $types(x, \Gamma) = \{\tau_1, \ldots, \tau_n\}$, if $x : \tau_1 \cap \ldots \cap \tau_n \in \Gamma$
2. $types(x, \Gamma) = \emptyset$, otherwise

Definition 24
Given two type environments Γ_1 and Γ_2, $\Gamma_1 - \Gamma_2$ is the new environment where, for each variable x in Γ_1, $x : \tau_1 \cap \ldots \cap \tau_n \in \Gamma_1 - \Gamma_2$ where

$$\{\tau_1, \ldots, \tau_n\} = types(x, \Gamma_1) \setminus types(x, \Gamma_2)$$

($\setminus$ stands for the standard set difference)

Lemma 10
If $\Gamma_M \vdash M[N/x] : \tau$ and $\Gamma_N \vdash N : \sigma$ for some type σ and environment Γ_N, then there is an environment Γ such that:

1. $\Gamma_M \subseteq \Gamma$,
2. $\Gamma \vdash (\lambda x.M)N : \tau$.

Proof
We may assume that x does not occur in Γ_M. We will consider two cases:

1. $x \in FV(M)$. Let $\Gamma = \Gamma_M$. By obvious type derivations the statement follows from the following claim:
   There are types $\sigma_1, \ldots, \sigma_n$ and environments $\Gamma_1, \ldots, \Gamma_n$ such that:
   (a) $\Gamma_1 \vdash N : \sigma_1, \ldots, \Gamma_n \vdash N : \sigma_n$,
   (b) $(\Gamma - (\Gamma_1 \wedge \ldots \wedge \Gamma_n)) \wedge \{x : \sigma_1 \cap \ldots \cap \sigma_n\} \vdash M : \tau$.
   The claim is proved by induction on $(size(M), size(\tau))$.
   (a) M is a variable x. The assumption is $\Gamma \vdash N : \tau$. Take $n = 1$, $\sigma_1 = \tau$ and $\Gamma_1 = \Gamma$.
   (b) M is of the form $\lambda y.P$ (with $y \notin FV(N)$). In this case $\tau = \tau_1 \to \tau_2$ and

$$\Gamma \wedge \{y : \tau_1\} \vdash P[N/x] : \tau_2$$

   By induction we have:
   i $\exists_{\sigma_1, \ldots, \sigma_n, \Gamma_1, \ldots, \Gamma_n} . (\Gamma_1 \vdash N : \sigma_1, \ldots, \Gamma_n \vdash N : \sigma_n)$
   ii $\Gamma - (\Gamma_1 \wedge \ldots \wedge \Gamma_n) \wedge \{x : \sigma_1 \cap \ldots \cap \sigma_n, y : \tau_1\} \vdash P : \tau_2$
   Then, by the rule $ABS - I$:

$$\Gamma - (\Gamma_1 \wedge \ldots \wedge \Gamma_n) \wedge \{x : \sigma_1 \cap \ldots \cap \sigma_n\} \vdash \lambda y.P : \tau_1 \to \tau_2$$

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

(c) M is of the form $M_1 M_2$. In this case we have:

i $\Gamma = \Gamma_0 \wedge \Gamma_1 \wedge \ldots \wedge \Gamma_n$

ii $\Gamma_0 \vdash M_1[N/x] : \tau_1 \cap \ldots \cap \tau_n \to \tau$

iii $\Gamma_i \vdash M_2[N/x] : \tau_i$ for $(1 \leqslant i \leqslant n)$

Let n_1 and n_2 be the number of occurrences of x in M_1 and M_2, respectively. By induction we have:

i $\exists_{\sigma_1,\ldots,\sigma_{n_1},\Gamma_1^*,\ldots,\Gamma_{n_1}^*}.(\Gamma_1^* \vdash N : \sigma_1, \ldots, \Gamma_{n_1}^* \vdash N : \sigma_{n_1})$

ii $\exists_{\sigma'_{i1},\ldots,\sigma'_{in_2},\Gamma'_{i1},\ldots,\Gamma'_{in_2}}.(\Gamma'_{i1} \vdash N : \sigma'_{i1}, \ldots, \Gamma'_{in_2} \vdash N : \sigma'_{in_2})$, for $(1 \leqslant i \leqslant n)$

iii $\Gamma_0 - (\Gamma_1^* \wedge \ldots \wedge \Gamma_{n_1}^*) \wedge \{x : \sigma_1 \cap \ldots \cap \sigma_{n_1}\} \vdash M_1 : \tau_1 \cap \ldots \cap \tau_n \to \tau$

iv $\Gamma_i - (\Gamma'_{i1} \wedge \ldots \wedge \Gamma'_{in_2}) \wedge \{x : \sigma'_{i1} \cap \ldots \cap \sigma'_{in_2}\} \vdash M_2 : \tau_i$, for $(1 \leqslant i \leqslant n)$

Now let $\bar{\sigma} = \sigma_1 \cap \ldots \cap \sigma_{n_1}$ and $\bar{\sigma}_i = \sigma'_{i1} \cap \ldots \cap \sigma'_{in_2}$. Let $\bar{\Gamma} = \Gamma_1^* \wedge \ldots \wedge \Gamma_{n_1}^*$ and $\bar{\Gamma}_i = \Gamma'_{i1} \wedge \ldots \wedge \Gamma'_{in_2}$ for $(1 \leqslant i \leqslant n)$. By the APP rule we have:

$$\Gamma - (\bar{\Gamma} \wedge \bar{\Gamma}_1 \wedge \ldots \wedge \bar{\Gamma}_n) \wedge \{x : \bar{\sigma} \cap \bar{\sigma}_1 \cap \ldots \cap \bar{\sigma}_n\} \vdash M_1 M_2 : \tau$$

2. $x \notin FV(M)$. In this case $\Gamma_M \vdash M[N/x] : \tau$ is $\Gamma_M \vdash M : \tau$. By hypothesis, there are Γ_N and σ such that $\Gamma_N \vdash N : \sigma$. Let $\Gamma = \Gamma_M \wedge \Gamma_N$. By application of the $ABS - K$ and APP rules we have:

$$\Gamma \vdash (\lambda x.M)N : \tau$$

□

Theorem 7
If M is strongly normalizable, then there are Γ and σ such that $\Gamma \vdash M : \sigma$.

Proof
We will use induction on $(depth(M), size(M))$. Note that any λ-term has exactly one of the following forms:

1. $\lambda x_1 \ldots x_n.xM_1 \ldots M_p$, where x may or may not be one of the $x_i's$, $(n \geqslant 0, p \geqslant 0)$,
2. $\lambda x_1 \ldots x_n.(\lambda x.M)M_1 \ldots M_p$ $(n \geqslant 0, p \geqslant 1)$.

We now proceed by cases:

1. $M \equiv \lambda x_1 \ldots x_n.xM_1 \ldots M_p$, with $x \neq x_1, \ldots, x_n$. By induction we have:

$$\Gamma_1 \vdash M_1 : \sigma_1, \ldots, \Gamma_p \vdash M_p : \sigma_p$$

By successive applications of the APP rule, we have:

$$\Gamma_1 \wedge \ldots \wedge \Gamma_p \wedge \{x : \sigma_1 \to \ldots \to \sigma_p \to \sigma\} \vdash xM_1 \ldots M_p : \sigma$$

Let $x_i : \tau_i$ be the type assignment for x_i in $\Gamma_1 \wedge \ldots \wedge \Gamma_p$ and let Γ be $\Gamma_1 \wedge \ldots \wedge \Gamma_p$ where we remove the type assignments $x_i : \tau_i$, for $(1 \leqslant i \leqslant n)$. Then, by successive applications of the $ABS - K$ and the $ABS - I$ rules, we have:

$$\Gamma \vdash \lambda x_1 \ldots x_n.xM_1 \ldots M_p : \tau_1 \to \ldots \to \tau_n \to \sigma$$

where τ_i is the type declared for x_i in $\Gamma_1 \wedge \ldots \wedge \Gamma_p$ if such declaration exists, or a fresh type variable if it does not exist.

https://doi.org/10.1017/S0956796803004970 Published online by Cambridge University Press

2. $M \equiv \lambda x_1 \ldots x_n.x_i M_1 \ldots M_p$. Similar to the previous case.
3. $M \equiv \lambda x_1 \ldots x_m.(\lambda x.N)P N_1 \ldots N_n$. By induction:

$$\Gamma_1 \vdash P : \sigma$$

and

$$\Gamma_2 \vdash N[P/x]N_1 \ldots N_n : \tau$$

We claim that there is a type environment Γ' such that $\Gamma_2 \subseteq \Gamma'$ and $\Gamma' \vdash (\lambda x.N)P N_1 \ldots N_n : \tau$. Let Γ be Γ' where we remove the type assignments $x_i : \tau_i$, for $(1 \leqslant i \leqslant m)$. By successive applications of the rules $ABS-K$ and $ABS-I$ we have:

$$\Gamma \vdash M : \tau_1 \to \ldots \to \tau_m \to \tau$$

where each τ_i is the type declared for x_i in Γ', if such declaration exists, or a fresh variable if it does not exist. Our claim is proved by induction on n where the base case is lemma 10.

□

Theorem 8
If M is such that there is an environment Γ and a type σ such that $\Gamma \vdash M : \sigma$, then M is typable in the Coppo-Dezani type system ((Coppo & Dezani-Ciancaglini, 1980)).

Proof
Easy by induction in the derivation tree for $\Gamma \vdash M : \sigma$. □

Proof of Theorem 1
It follows directly from Theorems 7 and 8. □

References

Abramsky, S. (1993) Computational interpretations of linear logic. *Theor. Comput. Sci.* **111**, 3–57.

Amadio, R. M. and Curien, P.-L. (1998) *Domains and Lambda-Calculi*. Cambridge University Press.

Barendregt, H. (1992) Lambda calculi with types. In: Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (editors), *Handbook of Logic in Computer Science, vol. 2*, pp. 117–309. Oxford Science Publications.

Barendregt, H., Coppo, M. and Dezani-Ciancaglini, M. (1983) A filter lambda model and the completeness of type assignement. *J. Symbolic Logic*, **48**(4), 931–940.

Benton, N., Bierman, G., de Paiva, V. and Hyland, M. (1992) *Term assignement for intuitionistic linear logic*. Technical report, Computing Laboratory, University of Cambridge.

Bucciarelli, A., Lorenzis, S. De, Piperno, A. and Salvo, I. (1999) Some computational properties of intersection types. *14th IEEE Symposium on Logic in Computer Science*.

Coppo, M. and Dezani-Ciancaglini, M. (1980) An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, **21**(4), 685–693.

Coppo, M. and Giannini, P. (1995) Principal types and unification for simple intersection type systems. *Infor. & Computation*, **122**(1).

Curry, H. B. (1934) Functionality in combinatory logic. *Proc. Nat. Acad. Sci. U.S.A.*, **20**, 584–590.

Damas, L. and Milner, R. (1982) Principal type schemes for functional languages. *9th ACM Symposium on Principles of Programming Languages.*

Damiani, F. and Giannini, P. (1994) A decidable intersection type system based on relevance. *Theoretical Aspects of Computer Science. Lecture Notes in Computer Science*. Springer-Verlag.

Fradet, P. (1994) Compilation of head and strong reduction. *Proceedings 5th European Symposium on Programming. Lecture Notes in Computer Science 788.*

Hindley, R. (1997) *Basic Simple Type Theory*. Cambridge University Press.

Jim, T. (1996) What are principal typings and are they good for? *ACM Symposium on Principles of Programming Languages.*

Kfoury, A. J. (1996) Beta-reduction as unification. *Logic, Algebra and Computer Science (H. Rasiowa Memorial Conference).*

Kfoury, A. J. (2000) A linearization of the lambda-calculus. *J. Logic & Computation* **10**(3), 411–436.

Kfoury, A. J. and Wells, J. B. (1999) Principality and decidable type inference for finite-rank intersection types. *Conference Record, POPL'99: 26th ACM Symposium on Principles of Programming Languages.*

Lincoln, P. and Mitchell, J. (1992) Operational aspects of linear lambda calculus. *7th Symposium on Logic in Computer Science.*

Peyton Jones, S. L. (1987) *The Implementation of Functional Programming Languages.* Prentice Hall International.

van Bakel, S. (1993) *Intersection type disciplines in lambda calculus and applicative term rewriting systems.* PhD thesis, Department of Computer Science, University of Nijmegen.

Wadler, P. (1990) Linear types can change the world! In: Broy, M. and Jones, C. (editors), *Programming Concepts and Methods.* North Holland.