Semantics with Intersection Types*

Steffen van Bakel

Department of Computing, Imperial College of Science, Technology and Medicine,
180 Queen's Gate, London SW7 2BZ, U.K.,
E-mail: svb@doc.ic.ac.uk

(Sections 4 through 7 are based on a paper co-authored by
Maribel Fernández, of King's College, London.)

Contents

*These notes contain material that appeared before, in slightly different form, in [1], [3], and [9]

Introduction

In the recent years several notions of type assignment for several (extended) lambda calculi have been studied. The oldest among these is a well understood and elegantly defined notion of type assignment on lambda terms, known as the Curry type assignment system [18]. It expresses abstraction and application, and can be used to obtain a (basic) functional characterization of terms. It is well known that in that system, the problem of typeability

Given a term M, are there a basis B and a type σ such that $B \vdash M : \sigma$?

is decidable, and that it has the principal type property:

If M is typeable, then there are P, π such that $P \vdash M : \pi$, and, for every B, σ such that $B \vdash M : \sigma$, there exist a way of generating $\langle B, \sigma \rangle$ from $\langle P, \pi \rangle$.

These two properties found their way into programming, mainly through the pioneering work of R. Milner [36]. He introduced a functional programming language ML, of which the underlying type system is an extension of Curry's system. The extension consists of the introduction of polymorphic functions, i.e. functions that can be applied to various kinds of arguments, even of incomparable type. The formal motivation of this concept lies directly in the notion of principal types.

Though the Curry system is already powerful and convenient for use in programming practice, it has drawbacks. It is, for example, not possible to assign a type to the term $(\lambda x.xx)$, and terms that are β-equal can have different principal type schemes. The Intersection Type Discipline as presented in [14] by M. Coppo, M. Dezani-Ciancaglini, and B. Venneri (a more enhanced system was presented in [11] by H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini) is an extension of Curry's system that does not have these drawbacks. The extension being made consists mainly of allowing for term variables (and terms) to have more than one type. Intersection types are constructed by adding, next to the type constructor '$\rightarrow$' of Curry's system, the type constructor '$\cap$' and the type constant 'ω'. This slight generalization causes a great change in complexity; in fact, now all terms having a (head) normal form can be characterized by their assignable types, a property that immediately shows that type assignment (even in the system that does not contain ω, see [1]) is undecidable. Also, by introducing this extension a system is obtained that is closed under β-equality: if $B \vdash M : \sigma$ and $M =_\beta N$, then $B \vdash N : \sigma$.

The type assignment system presented in [11] (the BCD-system) is based on the system as presented in [14]. It defines the set of intersection types in a more general way by treating '$\cap$' as a general type constructor, and introduces two derivation rules for introduction and elimination of intersections; the handling of intersection in this way is inspired by the similarity between intersection and logical conjunction. A big contribution of [11] to the theory of intersection types is the introduction of a filter λ-model and the proof of completeness of type assignment; to achieve the latter, the system is strengthened further by introducing a partial order relation '$\leq$' on types as well as adding the type assignment rule $(\leq)$.

A disadvantage of the BCD-system (and of any real intersection system, for that matter) is that type assignment in this system is undecidable. In recent years, some decidable restrictions have been studied. The first was the Rank2 intersection type assignment system [4], as first suggested by D. Leivant in [34], that is very close to the notion of type assignment as used in ML. The key idea for this system is to restrict the set of types to those of the shape $((\sigma_1 \cap \cdots \cap \sigma_n) \rightarrow \tau)$, where the σ_i $(i \in \underline{n})$ are types that do not contain intersections.

That intersection types can be used as a basis for programming languages was first discussed by J. Reynolds in [40]. This led to the development of the (typed) programming language Forsythe [41], and to the work of B.C. Pierce [38, 39], who studied intersection types and bounded polymorphism in the field of typed lambda calculi. Because there only typed systems are considered, the systems are decidable.

Another disadvantage of the BCD-system is that it is too general: in this system there are several ways to deduce a desired result, due to the presence of the derivation rules $(\cap I)$, $(\cap E)$ and $(\leq)$. These rules not only allow of superfluous steps in derivations, but also make it possible to give essentially different derivations for the same result. Moreover, in [11] the relation $\leq$ induced an equivalence relation $\sim$ on types. Equivalence classes are big (for example: $\omega \sim (\sigma \to \omega)$, for all types σ) and type assignment is closed for $\sim$.

The BCD-system has the principal type property, as was shown in [42]; although for every M the set $\{\langle B, \sigma\rangle \mid B \vdash M : \sigma\}$ can be generated using operations specified in [42], the problem of type-checking

Given a term M and type σ, is there a basis B such that $B \vdash M : \sigma$?

is complicated. This is not only due to the undecidability of the problem, but even a semi-algorithm is difficult to define, due to the equivalence relation on types. Moreover, because of the general treatment of intersection types, the sequence of operations needed to go from one type to another is normally not unique.

The strict type assignment system as defined in [1] is a restriction of the system of [11]; it uses a set of strict types, that is actually the set of normalized tail-proper types of [14]. Although there are rather strong restrictions imposed, the provable results for the strict system are very close to those for the system of [11]. For example, the sets of normalizable terms and those having a normal form can be equally elegantly characterized. The main difference between the two systems is that the strict system is *not* closed for η-reduction, whereas the BCD-system is.

The strict system gives rise to a strict filter λ-model that satisfies all major properties of the filter λ-model as presented in [11], but is an essentially different λ-model, equivalent to Engeler's model $\mathcal{D}_A$ [22]. In [1] was shown that soundness for the notion of type assignment of [11] is lost if instead of simple type semantics, the inference type semantics is used. With the use of the inference type semantics, in [1] soundness and completeness for strict type assignment was proved, without having the necessity of introducing $\leq$.

The set of types assignable to a term M in the strict system is significantly smaller than the set of types assignable to M in the BCD-system. In particular, the problem of type checking for the strict system is, because of the smaller equivalence classes, less complicated than for the BCD-system.

The type assignment system as presented here was first presented in [3] (albeit in different notation), and is a true restriction of the BCD-system that satisfies all properties of that system, and is also an extension of Curry's system. It will be shown that, in order to prove a completeness result using intersection types, there is no need to be as general as in [11]; this result can also be obtained for the system presented here. The main advantage of this system over the BCD-system is that the set of types assignable to a term is significantly smaller. An other advantage of the system is that derivations are syntax-directed: there is, unlike in the BCD-system, a one-one relationship between terms and skeletons of derivations. These two features are supported by a less complicated type structure.

The system presented here is also an extension of the strict type assignment system as presented in [1]. The major difference is that the system will prove to be closed for η-reduction: If $B \vdash M:\sigma$ and $M \to_\eta N$, then $B \vdash N:\sigma$. This does not hold for the strict system.

Some results already known for, for example, the BCD-system, hold for the system $\vdash$.

- If $B \vdash M:\sigma$ and $M \to_\eta N$, then $B \vdash N:\sigma$.
- If $B \vdash M:\sigma$ and $M =_\beta N$, then $B \vdash N:\sigma$.
- $B \vdash M:\sigma$ and $\sigma \neq \omega$, if and only if M has a head normal form.
- $B \vdash M:\sigma$ and ω does not occur in B and σ, if and only if M has a normal form.
- $\vdash$ has the principal type property.

The first four of these properties will be reviewed here; for the last, see [3].

Notations

In these notes, the symbol φ will be a type-variable; Greek symbols like α, β, μ, ρ, σ, and τ will range over types, and π will be used for principal types. '$\to$' will be assumed to associate to the right, and '$\cap$' binds stronger than '$\to$'. M, N are used for lambda terms; C, D, E for (arbitrary) combinators, $\mathsf{C}, \mathsf{D}, \mathsf{E}$ for concrete combinators, and t, u, v for terms in Combinator Systems; x, y, z for term-variables, $M[N/x]$ for the usual operation of substitution on lambda terms, A for terms in $\Lambda\bot$-normal form, and a for approximants of combinator systems. B is used for bases, $B\backslash x$ for the basis obtained from B by erasing the statement that has x as subject, and P for principal bases. All symbols can appear indexed.

We will write $\underline{n}$ for the set $\{1, \ldots, n\}$, and will often use a vector notation '$\vec{\cdot}$' for the purpose of abbreviation. For example, $P\vec{M_i}$ stands for $PM_1 \cdots M_n$ for a suitable n, and $[N_1/x_1, \ldots, N_n/x_n]$ is abbreviated by $[\vec{N_i}/\vec{x_i}]$.

Two types (bases, pairs of basis and type) are *disjoint* if and only if they have no type-variables in common. Notions of type assignment are defined as ternary relations on bases, terms, and types, that are denoted by $\vdash$, possibly indexed if necessary. If in a notion of type assignment for M there are basis B and type σ such that $B \vdash M:\sigma$, then M is *typed with* σ, and σ is *assigned to* M.

1 Type assignment

In this section a notion of type assignment system is presented that is a restricted version of the BCD-system presented in [11], together with some of its properties. The major feature of this restricted system is, compared to the BCD-system, a restricted version of the derivation rules and the use of strict types. It also forms a slight extension of the strict type assignment system that was presented in [1]; the main difference is that the strict system is not closed for η-reduction, whereas the system presented here is.

Strict types are the types that are strictly needed to assign a type to a term in the BCD-system. In the set of strict types, intersection type schemes and the type constant ω play a limited role. In particular, ω is taken to be the empty intersection: if $n = 0$, then $\cap_{\underline{n}}\sigma_i \equiv \omega$, so ω does not occur in an intersection subtype. Moreover, intersection type schemes (so also ω) occur in strict types only as subtypes at the left-hand side of an arrow type scheme, as in the types of [12], [13], and [14].

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

$$\begin{array}{rcl} \mathcal{T}_s & ::= & \varphi \mid (\mathcal{T} \to \mathcal{T}_s) \\ \mathcal{T} & ::= & (\mathcal{T}_s \cap \cdots \cap \mathcal{T}_s) \end{array}$$

We will write ω for an intersection of zero strict types, and $\cap_{\underline{n}}\sigma_i$ for the type $\sigma_1 \cap \cdots \cap \sigma_n$; we will also, as usual, omit right-most, outer-most brackets.

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

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

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

iv) On $\mathcal{T}$, the relation $\sim$ is defined by: $\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$.

Notice that $\sigma \leq \omega$, for all σ. Unless stated otherwise, if a type is written as $\cap_{\underline{n}}\sigma_i$, then all $\sigma_i\,(i \in \underline{n})$ are assumed to be strict.

For the relation $\leq$, the following properties hold:

Lemma 1.2 *i)* $\sigma \leq \tau \Rightarrow \sigma \leq \tau$.

ii) $\varphi \leq \sigma \iff \sigma \equiv \varphi$. *So* $\{\sigma \mid \sigma \sim \varphi\} = \{\varphi\}$.

iii) $\omega \leq \sigma \iff \sigma \equiv \omega$. *So* $\{\sigma \mid \sigma \sim \omega\} = \{\omega\}$.

iv) $\sigma \to \tau \leq \rho \in \mathcal{T}_s \iff \exists\, \alpha \in \mathcal{T}, \beta \in \mathcal{T}_s\, [\rho \equiv \alpha \to \beta \;\&\; \alpha \leq \sigma \;\&\; \tau \leq \beta]$.

v) $\cap_{\underline{n}}\sigma_i \leq \tau \in \mathcal{T}_s \Rightarrow \exists\, i \in \underline{n}\, [\sigma_i \leq \tau]$.

vi) $\sigma \leq \tau \Rightarrow \exists \sigma_i\,(i \in \underline{n}), \tau_j\,(j \in \underline{m})\, [\sigma = \cap_{\underline{n}}\sigma_i \;\&\; \tau = \cap_{\underline{m}}\tau_j \;\&\; \forall\, j \in \underline{m}\; \exists i \in \underline{n}\, [\sigma_i \leq \tau_j]]$.

Proof: Easy. ■

Definition 1.3 (BASES) *i*) A *basis* is a partial mapping from term variables to types, normally written as a set of statements of the shape $x{:}\sigma$.

ii) If $B_i\,(i \in \underline{n})$ are bases, then $\bigcap\{B_1, \ldots, B_n\}$ (or $\bigcap\{\underline{n}\}B_i$) is the basis defined as follows: $x{:}\cap_{\underline{m}}\sigma_i \in \bigcap\{\underline{n}\}B_i$ if and only if $\{x{:}\sigma_1, \ldots, x{:}\sigma_m\}$ is the set of all statements about x that occur in $B_1 \cup \ldots \cup B_n$.

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

Often $B \cup \{x{:}\sigma\}$ (or $B, x{:}\sigma$) will be written for the basis $\bigcap\{B, \{x{:}\sigma\}\}$, when x does not occur in B.

Definition 1.4 (TYPE ASSIGNMENT) *Type assignment* and *derivations* are defined by the following natural deduction system (where all types displayed are strict, except for σ in the rule

(*Ax*), ($\rightarrow$*E*) and ($\rightarrow$*I*)):

$$(Ax): \frac{}{B, x{:}\sigma \vdash x{:}\tau}\ (\sigma \leq \tau \in \mathcal{T}_s) \qquad (\cap I): \frac{B \vdash M{:}\sigma_1 \quad \cdots \quad B \vdash M{:}\sigma_n}{B \vdash M{:}\cap_{\underline{n}}\sigma_i}\ (n \geq 0)$$

$$(\rightarrow I): \frac{B, x{:}\sigma \vdash M{:}\tau}{B \vdash \lambda x.M{:}\sigma{\rightarrow}\tau}\ (\sigma \in \mathcal{T}) \qquad (\rightarrow E): \frac{B \vdash M{:}\sigma{\rightarrow}\tau \quad B \vdash N{:}\sigma}{B \vdash MN{:}\tau}$$

$B \vdash M{:}\sigma$ is used if this statement is derivable using a strict intersection derivation, and D :: $B \vdash M{:}\sigma$ specifies that this result was obtained through the derivation D.

For this notion of type assignment, the following properties hold:

Lemma 1.5 *i)* $B \vdash x{:}\sigma \iff \exists \rho \in \mathcal{T}\ [x{:}\rho \in B\ \&\ \rho \leq \sigma]$.
ii) $B \vdash MN{:}\sigma\ \&\ \sigma \in \mathcal{T}_s \iff \exists \tau \in \mathcal{T}\ [B \vdash M{:}\tau{\rightarrow}\sigma\ \&\ B \vdash N{:}\tau]$.
iii) $B \vdash \lambda x.M{:}\sigma \iff \exists \rho \in \mathcal{T}, \mu \in \mathcal{T}_s\ [\sigma = \rho{\rightarrow}\mu\ \&\ B, x{:}\rho \vdash M{:}\mu]$.
iv) $B \vdash M{:}\sigma\ \&\ \sigma \in \mathcal{T} \iff \exists \sigma_i\ (i \in \underline{n})\ [\sigma = \cap_{\underline{n}}\sigma_i\ \&\ \forall i \in \underline{n}\ [B \vdash M{:}\sigma_i]]$.
v) $B \vdash M{:}\sigma \iff \{x{:}\tau \in B \mid x \in \mathit{fv}(M)\} \vdash M{:}\sigma$.
vi) $B \vdash M{:}\sigma\ \&\ B' \leq B \Rightarrow B' \vdash M{:}\sigma$.
vii) $B \vdash M{:}\sigma \Rightarrow \{x{:}\rho \mid x{:}\rho \in B\ \&\ x \in \mathit{fv}(M)\} \vdash M{:}\sigma$.

Proof: Easy. ■

1.1 Eta reduction

Although the rule (*Ax*) is defined only for term-variables, $\vdash$ is closed for $\leq$ and weakening.

Lemma 1.6 (WEAKENING) *If $B \vdash M{:}\sigma$ and $B' \leq B, \sigma \leq \tau$, then $B' \vdash M{:}\tau$, so the following is an admissible rule in $\vdash$:*

$$(\leq): \frac{B \vdash M{:}\sigma}{B' \vdash M{:}\tau}\ (B' \leq B, \sigma \leq \tau)$$

Proof: By induction on $\vdash$.
(*Ax*) : Then $M \equiv x$, and there is $x{:}\rho \in B$ such that $\rho \leq \sigma$. Since $B' \leq B$, there is $x{:}\mu \in B'$ such that $\mu \leq \rho$. Notice that $\mu \leq \rho \leq \sigma \leq \tau$, so, by Lemma *1.5 (i)*, $B' \vdash x{:}\tau$.
($\rightarrow$*I*) : Then $M \equiv \lambda x.M'$, and there are $\rho \in \mathcal{T}, \mu \in \mathcal{T}_s$ such that $\sigma = \rho{\rightarrow}\mu$ and $B, x{:}\rho \vdash M'{:}\mu$. By Lemma *1.2 (vi)* & *(iv)* there are $\rho_i, \mu_i\ (i \in \underline{n})$ such that $\tau = \cap_{\underline{n}}(\rho_i{\rightarrow}\mu_i)$, and for $i \in \underline{n}$, $\rho_i \leq \rho$ and $\mu \leq \mu_i$. Since $B' \leq B$ and $\rho_i \leq \rho$, also $B', x{:}\rho_i \leq B, x{:}\rho$, and by induction $B, x{:}\rho_i \vdash M'{:}\mu_i$. So, by ($\rightarrow$*I*), for every $i \in \underline{n}$, $B \vdash \lambda x.M'{:}\rho_i{\rightarrow}\mu_i$, so, by ($\cap$*I*), $B \vdash \lambda x.M'{:}\tau$.
($\rightarrow$*E*) : Then $M \equiv M_1M_2$ and there is a $\mu \in \mathcal{T}$ such that $B \vdash M_1{:}\mu{\rightarrow}\sigma$ and $B \vdash M_2{:}\mu$. Since $\sigma \leq \tau$, also $\mu{\rightarrow}\sigma \leq \mu{\rightarrow}\tau$ and, by induction, $B \vdash M_1{:}\mu{\rightarrow}\tau$. Then, by ($\rightarrow$*E*), $B \vdash M_1M_2{:}\tau$. ■
($\cap$*I*) : Then $\sigma = \cap_{\underline{n}}\sigma_i$, and, for every $i \in \underline{n}$, $B \vdash M{:}\sigma_i$. By Lemma *1.2 (vi)*, there are $\tau_j\ (j \in \underline{m})$ such that $\tau = \cap_{\underline{m}}\tau_j$ and, for every $j \in \underline{m}$, there is a $i \in \underline{n}$ such that $\sigma_i \leq \tau_j$. By induction, for every $j \in \underline{m}$, $B' \vdash M{:}\tau_j$. But then, by ($\cap$*I*), $B' \vdash M{:}\tau$.

Now it is easy to prove that type assignment in this system is closed under η-reduction. The proof for this result is split in two parts, Lemma *1.7* and Theorem *1.8*. The lemma is also used in the proof of Lemma *2.7*.

Lemma 1.7 *If* $\tau \in \mathcal{T}_s$, $B, x{:}\sigma \vdash Mx{:}\tau$ *and* $x \notin fv(M)$ *then* $B \vdash M{:}\sigma{\to}\tau$.

Proof:
$$\begin{array}{ll}
\tau \in \mathcal{T}_s \;\&\; B, x{:}\sigma \vdash Mx{:}\tau \;\&\; x \notin fv(M) \;\Rightarrow & (\to E)\\
\exists \mu\, [B, x{:}\sigma \vdash M{:}\mu{\to}\tau \;\&\; B, x{:}\sigma \vdash x{:}\mu \;\&\; x \notin fv(M)] \;\Rightarrow & (1.5(i))\\
\exists \mu\, [B, x{:}\sigma \vdash M{:}\mu{\to}\tau \;\&\; \sigma \leq \mu \;\&\; x \notin fv(M)] \;\Rightarrow & (1.5(vii))\\
\exists \mu\, [B \vdash M{:}\mu{\to}\tau \;\&\; \sigma \leq \mu] \;\Rightarrow & (1.1(iii))\\
\exists \mu\, [B \vdash M{:}\mu{\to}\tau \;\&\; \mu{\to}\tau \leq \sigma{\to}\tau] \;\Rightarrow & (1.6)\\
B \vdash M{:}\sigma{\to}\tau. & \blacksquare
\end{array}$$

Theorem 1.8 ($\vdash$ CLOSED FOR η-REDUCTION) *If* $B \vdash M{:}\sigma$ *and* $M \to_\eta N$, *then* $B \vdash N{:}\sigma$.

Proof: By induction on the definition of $\to_\eta$, of which only the part $\lambda x.Mx \to_\eta M$ is shown, where x does not occur free in M. The other parts are dealt with by straightforward induction.

$(\sigma \in \mathcal{T}_s)$: Then:
$$\begin{array}{ll}
B \vdash \lambda x.Mx{:}\sigma \;\&\; x \notin fv(M) \;\Rightarrow & (\to I)\\
\exists \rho, \mu\, [\sigma = \rho{\to}\mu \;\&\; B, x{:}\rho \vdash Mx{:}\mu] \;\Rightarrow & (1.7)\\
B \vdash M{:}\sigma.
\end{array}$$

$(\sigma = \cap_{\underline{n}}\sigma_i)$: Then, by $(\cap I)$, $B \vdash \lambda x.Mx{:}\sigma_i$ for all $i \in \underline{n}$, so, by the previous part, $B \vdash M{:}\sigma_i$, so, by $(\cap I)$, $B \vdash M{:}\sigma$. $\blacksquare$

By the structure of this proof, below we will normally focus on strict types when proving properties.

For example, $\emptyset \vdash \lambda xy.xy{:}(\sigma{\to}\tau){\to}\sigma{\cap}\rho{\to}\tau$ and $\emptyset \vdash \lambda x.x{:}(\sigma{\to}\tau){\to}\sigma{\cap}\rho{\to}\tau$ are both easy to derive.

$$\dfrac{\dfrac{\dfrac{}{x{:}\sigma{\to}\tau, y{:}\sigma{\cap}\rho \vdash x{:}\sigma{\to}\tau} \qquad \dfrac{}{x{:}\sigma{\to}\tau, y{:}\sigma{\cap}\rho \vdash y{:}\sigma}\,(\sigma{\cap}\rho \leq \sigma)}{x{:}\sigma{\to}\tau, y{:}\sigma{\cap}\rho \vdash xy{:}\tau}}{x{:}\sigma{\to}\tau \vdash \lambda y.xy{:}\sigma{\cap}\rho{\to}\tau}}{\emptyset \vdash \lambda xy.xy{:}(\sigma{\to}\tau){\to}\sigma{\cap}\rho{\to}\tau}$$

$$\dfrac{x{:}\sigma{\to}\tau \vdash x{:}\sigma{\cap}\rho{\to}\tau}{\emptyset \vdash \lambda x.x{:}(\sigma{\to}\tau){\to}\sigma{\cap}\rho{\to}\tau}\,(\sigma{\to}\tau \leq \sigma{\cap}\rho{\to}\tau)$$

1.2 Subject reduction and expansion

As in [13, 11, 1], it is possible to prove that the type assignment system is closed under $=_\beta$. In the latter two papers this result was obtained by building a filter λ-model; from the fact that every M is interpreted by the set of its assignable types, and that set is a filter, the result is then immediate (see also Corollary *3.13*). In this paper the result will first be obtained directly, without constructing a filter model; in this way the precise behaviour of the type constructor '$\cap$' and the type constant ω can be made apparent.

That the system is closed under subject reduction can be illustrated also by the following 'Cut and Paste' proof: Suppose that $B \vdash (\lambda x.M)N{:}\sigma$, with $\sigma \in \mathcal{T}_s$. By $(\to E)$, there exists τ

such that

$$B \vdash \lambda x.M : \tau \to \sigma \text{ and } B \vdash N : \tau.$$

Since $(\to I)$ should be the last step performed for the first result, also

$$B, x{:}\tau \vdash M : \sigma \text{ and } B \vdash N : \tau.$$

Now there are (strict) types ρ_j $(j \in \underline{m})$ such that, for every ρ_j, in the first derivation, there exists a sub-derivation of the shape

$$\frac{}{B, x{:}\tau \vdash x : \rho_j}\ (Ax)$$

and these are all the applications of rule (Ax) that deal with x. Then, for all $j \in \underline{m}$, $\tau \leq \rho_j$ and, by Lemma *1.6*, $B \vdash N : \rho_j$. Then a derivation for $B \vdash M[N/x] : \sigma$ can be obtained by replacing, for every $j \in \underline{m}$, in the derivation for $B, x{:}\tau \vdash M : \sigma$, the sub-derivation $B, x{:}\tau \vdash x : \rho_j$ by the (new) derivation for $B \vdash N : \rho_j$.

The problem to solve in a proof for closure under β-equality is then that of β-expansion:

$$\text{if } B \vdash M[N/x] : \sigma, \text{ then } B \vdash (\lambda x.M)N : \sigma.$$

Assume that the term-variable x occurs in M and the term N is a sub-term of $M[N/x]$, so N is typed in the derivation for D :: $B \vdash M[N/x] : \sigma$, probably with several different types σ_i $(i \in \underline{n})$. A derivation for

$$B, x{:}\cap_{\underline{n}}\sigma_i \vdash M : \sigma$$

can be obtained by replacing, in D, all derivations for $B \vdash N : \sigma_i$ by the derivation for

$$\{x{:}\cap_{\underline{n}}\sigma_i\} \vdash x : \sigma_i.$$

Then, using $(\cap I)$, $B \vdash N : \cap_{\underline{n}}\sigma_i$, and, using $(\to I)$, $B \vdash \lambda x.M : \cap_{\underline{n}}\sigma_i \to \sigma$. Then, using $(\to E)$, the redex can be typed.

When the term-variable x does not occur in M, the term N is a not a sub-term of $M[N/x]$ and $B \vdash M[N/x] : \sigma$ stands for $B \vdash M : \sigma$. In this case, the type ω is used: since x does not occur in M, $x{:}\omega$ can be assumed to appear in B, and rule $(\to I)$ gives $B \vdash \lambda x.M : \omega \to \sigma$. By $(\cap I)$, $B \vdash N : \omega$, so, using $(\to E)$, the redex can be typed.

To show this result formally, first a substitution lemma is proved. Notice that, unlike for many other notions of type assignment (Curry's system, the polymorphic type discipline [24]), the implication holds in both directions.

Lemma 1.9 (SUBSTITUTION LEMMA) $\exists \rho\, [B, x{:}\rho \vdash M : \sigma \ \&\ B \vdash N : \rho] \iff B \vdash M[N/x] : \sigma$.

Proof: By induction on M. Only the case $\sigma \in \mathcal{T}_s$ is considered.

$(M \equiv x)$: $(\Rightarrow)$: $\exists \rho\, [B, x{:}\rho \vdash x : \sigma \ \&\ B \vdash N : \rho] \Rightarrow$ (*1.5(i)*)
$\exists \rho\, [\rho \leq \sigma \ \&\ B \vdash N : \rho] \Rightarrow$ (*1.6*)
$B \vdash x[N/x] : \sigma$.

$(\Leftarrow)$: $B \vdash x[N/x] : \sigma \Rightarrow B \vdash N : \sigma$; take $\rho = \sigma$.

$(M \equiv y \neq x)$: $(\Rightarrow)$: By Lemma *1.5(vii)*, since $y[N/x] \equiv y$.

$(\Leftarrow)$: $B \vdash y[N/x] : \sigma \Rightarrow B \vdash y : \sigma$; take $\rho = \omega$.

$$
\begin{array}{lll}
(M \equiv \lambda y.M')\,{:}\;(\Longleftrightarrow){:}\quad \exists \rho\,[B, x{:}\rho \vdash \lambda y.M{:}\sigma \;\&\; B \vdash N{:}\rho] & \Longleftrightarrow & (\rightarrow\!I) \\
\qquad \exists \rho, \alpha, \beta\,[B, x{:}\rho, y{:}\alpha \vdash M{:}\beta \;\&\; \sigma = \alpha{\rightarrow}\beta \;\&\; B \vdash N{:}\rho] & \Longleftrightarrow & (IH) \\
\qquad \exists \alpha, \beta\,[B, y{:}\alpha \vdash M[N/x]{:}\beta \;\&\; \sigma = \alpha{\rightarrow}\beta] & \Longleftrightarrow & (\rightarrow\!I) \\
\qquad B \vdash \lambda y.(M[N/x]){:}\sigma & \Longleftrightarrow & \\
\qquad B \vdash (\lambda y.M)[N/x]{:}\sigma. & &
\end{array}
$$

$$
\begin{array}{lll}
(M \equiv M_1 M_2)\,{:}\;(\Rightarrow){:}\quad \exists \rho\,[B, x{:}\rho \vdash M_1 M_2{:}\sigma \;\&\; B \vdash N{:}\rho] & \Longleftrightarrow & (\rightarrow\!E) \\
\qquad \exists \rho, \tau\,[B, x{:}\rho \vdash M_1{:}\tau{\rightarrow}\sigma \;\&\; B, x{:}\rho \vdash M_2{:}\tau \;\&\; B \vdash N{:}\rho] & \Longleftrightarrow & (IH) \\
\qquad \exists \tau\,[B \vdash M_1[N/x]{:}\tau{\rightarrow}\sigma \;\&\; B \vdash M_2[N/x]{:}\tau] & \Longleftrightarrow & (\rightarrow\!E) \\
\qquad B \vdash M_1[N/x]M_2[N/x]{:}\sigma & \Longleftrightarrow & \\
\qquad B \vdash (M_1 M_2)[N/x]{:}\sigma & &
\end{array}
$$

$$
\begin{array}{lll}
(\Leftarrow){:}\quad B \vdash M_1 M_2[N/x]{:}\sigma & \Rightarrow & \\
\qquad B \vdash M_1[N/x]M_2[N/x]{:}\sigma & \Longleftrightarrow & (\rightarrow\!E) \\
\qquad \exists \tau\,[B \vdash M_1[N/x]{:}\tau{\rightarrow}\sigma \;\&\; B \vdash M_2[N/x]{:}\tau] & \Longleftrightarrow & (IH) \\
\qquad \exists \rho_1, \rho_2, \tau\,[B, x{:}\rho_i \vdash M_1{:}\tau{\rightarrow}\sigma \;\&\; B \vdash N{:}\rho_1 \;\&\; B, x{:}\rho_2 \vdash M_2{:}\tau \;\&\; B \vdash N{:}\rho_2] & & \\
\qquad\qquad \Rightarrow\ (\rho = \rho_1{\cap}\rho_2 \;\&\; (\cap I) \;\&\; \textit{1.5(vi)}) & & \\
\qquad \exists \rho\,[B, x{:}\rho \vdash M_1 M_2{:}\sigma \;\&\; B \vdash N{:}\rho]. & & \blacksquare
\end{array}
$$

Theorem 1.10 ($\vdash$ CLOSED FOR $=_\beta$) $M =_\beta N \Rightarrow (B \vdash M{:}\sigma \Longleftrightarrow B \vdash N{:}\sigma)$, *so the following rule is an admissible rule in* $\vdash$*:*

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

Proof: By induction on the definition of $=_\beta$. The only part that needs attention is that of a redex, $B \vdash (\lambda x.M)N{:}\sigma \Longleftrightarrow B \vdash M[N/x]{:}\sigma$, where $\sigma \in \mathcal{T}_s$; all other cases follow by straightforward induction. To conclude, notice that, if $B \vdash (\lambda x.M)N{:}\sigma$, then, by $(\rightarrow\!E)$ and $(\rightarrow\!I)$, there exists a ρ such that $B, x{:}\rho \vdash M{:}\sigma$ and $B \vdash N{:}\rho$; the converse of this result holds, obviously, as well. The result then follows by applying Lemma *1.9*. ■

2 Approximation and normalization results

In [42] an approximation theorem is proved for the BCD-system, that formulates the relation between the types assignable to a term and those assignable to its approximants, as defined in [46] (see Definition *2.1* below):

$$B \vdash M{:}\sigma \textit{ if and only if there exists } A \in \mathcal{A}(M) \textit{ such that } B \vdash A{:}\sigma.$$

In this section, we will show this property for the system presented here. In [42] this result is obtained through a normalization of derivations, where all $(\rightarrow\!I)$–$(\rightarrow\!E)$ pairs, that derive a type for a redex $(\lambda x.M)N$, are replaced by one for its reduct $M[N/x]$, and all pairs of $(\cap I)$–$(\cap E)$ are eliminated. (This technique is also used in [13] and [11]. It requires a rather difficult notion of length of a derivation to show that this process terminates.) In this paper, the approximation theorem will be proved using the reducibility technique, following Tait [44], as was done in [15], and [20].

With this result, it can be shown that the BCD-system is conservative over the system presented here, and proven that the set of all terms having a (head) normal form are typeable in $\vdash$ (with a type without ω-occurrences) (Theorem *2.21*).

2.1 Approximants

The notion of approximant was first presented by C. Wadsworth [46] and is defined using the notion of terms in $\Lambda\bot$-normal form (like in [10], $\bot$ is used, instead of Ω; also, the symbol $\sqsubseteq$ is used as a relation on $\Lambda\bot$-terms, inspired by a similar relation defined on Böhm-trees in [10]).

Definition 2.1 (APPROXIMATE NORMAL FORMS) *i*) The set of $\Lambda\bot$ *-terms* is defined as the set Λ of lambda terms, extended by: $\bot \in \Lambda\bot$.

ii) The notion of reduction $\rightarrow_{\beta\bot}$ is defined as $\rightarrow_\beta$, extended by:

$$\begin{aligned} \lambda x.\bot &\rightarrow_{\beta\bot} \bot \\ \bot M &\rightarrow_{\beta\bot} \bot \end{aligned}$$

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

$$A ::= \bot \mid \lambda x.A \; (A \neq \bot) \mid x\overrightarrow{A_i} \; (n \geq 0)$$

The type assignment rules of the system are generalized to terms containing $\bot$ by allowing for the terms to be elements of $\Lambda\bot$. This implies that, because type assignment is almost syntax directed, if $\bot$ occurs in a term M and $B \vdash M:\sigma$, then either $\sigma = \omega$, or in the derivation for $M:\sigma$, $\bot$ appears in the right hand sub-term of an application, and this right-hand term is typed with ω. Moreover, the terms $\lambda x.\bot$ and $\bot\overrightarrow{M_i}$ are typeable by ω only.

Definition 2.2 (APPROXIMANTS) *i*) The partial order $\underset{\sim}{\sqsubseteq} \subseteq (\Lambda\bot)^2$ is defined as the transitive and reflexive closure of:

$$\begin{aligned} \bot &\underset{\sim}{\sqsubseteq} M \\ M \underset{\sim}{\sqsubseteq} M' &\Rightarrow \lambda x.M \underset{\sim}{\sqsubseteq} \lambda x.M' \\ M_1 \underset{\sim}{\sqsubseteq} M_1' \;\&\; M_2 \underset{\sim}{\sqsubseteq} M_2' &\Rightarrow M_1M_2 \underset{\sim}{\sqsubseteq} M_1'M_2' \end{aligned}$$

ii) For $A \in \mathcal{N}$, $M \in \Lambda$, if $A \underset{\sim}{\sqsubseteq} M$, then A is a *direct approximant* of M.

iii) The relation $\sqsubseteq \subseteq \mathcal{N} \times \Lambda$ is defined by: $A \sqsubseteq M \iff \exists M' =_\beta M \; [A \underset{\sim}{\sqsubseteq} M']$.

iv) If $A \sqsubseteq M$, then A is an *approximant* of M.

v) $\mathcal{A}(M) = \{A \in \mathcal{N} \mid A \sqsubseteq M\}$.

Lemma 2.3 $B \vdash M:\sigma \;\&\; M \underset{\sim}{\sqsubseteq} M' \Rightarrow B \vdash M':\sigma$.

Proof: By easy induction on the definition of $\underset{\sim}{\sqsubseteq}$; the base case, $\bot \underset{\sim}{\sqsubseteq} M'$, follows from the fact that then $\sigma = \omega$. ■

The following properties of approximants hold:

Lemma 2.4 *i) If* $A \in \mathcal{A}(x\overrightarrow{M_i})$ *and* $A' \in \mathcal{A}(N)$, *then* $AA' \in \mathcal{A}(x\overrightarrow{M_i}N)$.

ii) If $A \in \mathcal{A}(Mz)$ *and* $z \notin fv(M)$, *then either:*
- $A \equiv A'z$, $z \notin fv(A)$, *and* $A' \in \mathcal{A}(M)$, *or*
- $\lambda z.A \in \mathcal{A}(M)$.

iii) If $M =_\beta N$, *then* $\mathcal{A}(M) = \mathcal{A}(N)$.

Proof: Easy. ■

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

Definition 2.5 *i*) On $\Lambda\bot$, the partial mapping *join*, $\sqcup : \Lambda\bot \times \Lambda\bot \to \Lambda\bot$, is defined by:

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

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

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

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

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

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

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

Lemma 2.6 If $M_1 \sqsubseteq M$, and $M_2 \sqsubseteq M$, then $M_1 \sqcup M_2$ is defined, and $M_1 \sqsubseteq M_1 \sqcup M_2$, $M_2 \sqsubseteq M_1 \sqcup M_2$, and $M_1 \sqcup M_2 \sqsubseteq M$. ■

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

i) If $M_1 \equiv \bot$, then $M_1 \sqcup M_2 \equiv M_2$, so $M_1 \sqsubseteq M_1 \sqcup M_2$, $M_2 \sqsubseteq M_1 \sqcup M_2$, and $M_1 \sqcup M_2 \sqsubseteq M_2 \sqsubseteq M$. (The case $M_2 \equiv \bot$ goes similarly.)

ii) If $M_1 \equiv \lambda x.N_1$, then $M \equiv \lambda x.N$, $N_1 \sqsubseteq N$, and either $M_2 = \bot$ or $M_2 \equiv \lambda x.N_2$. The first case has been dealt with in part *(i)*, and for the other: then $N_2 \sqsubseteq N$. Then, by induction, $N_1 \sqsubseteq N_1 \sqcup N_2$, $N_2 \sqsubseteq N_1 \sqcup N_2$, and $N_1 \sqcup N_2 \sqsubseteq N$. Then also $\lambda x.N_1 \sqsubseteq \lambda x.N_1 \sqcup N_2$, $\lambda x.N_2 \sqsubseteq \lambda x.N_1 \sqcup N_2$, and $\lambda x.N_1 \sqcup N_2 \sqsubseteq \lambda x.N$. Notice that $\lambda x.N_1 \sqcup N_2 \equiv (\lambda x.N_1) \sqcup (\lambda x.N_2)$.

iii) If $M_1 \equiv P_1 Q_1$, then $M \equiv PQ$, $P_1 \sqsubseteq P$, $Q_1 \sqsubseteq Q$, and either $M_2 = \bot$ or $M_2 \equiv P_2 Q_2$. The first case has been dealt with in part *(i)*, and for the other: then $P_2 \sqsubseteq P$, $Q_2 \sqsubseteq Q$. By induction, we know $P_1 \sqsubseteq P_1 \sqcup P_2$, $P_2 \sqsubseteq P_1 \sqcup P_2$, and $P_1 \sqcup P_2 \sqsubseteq P$, as well as $Q_1 \sqsubseteq Q_1 \sqcup Q_2$, $Q_2 \sqsubseteq Q_1 \sqcup Q_2$, and $Q_1 \sqcup Q_2 \sqsubseteq Q$. Then also $P_1 Q_1 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, $P_2 Q_2 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, and $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \sqsubseteq PQ$. Notice that $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \equiv (P_1 Q_1) \sqcup (P_2 Q_2)$. ■

Notice that, because of *2.4(iii)*, $\mathcal{A}(M)$ can be used to define a semantics for the Lambda Calculus. In fact, it is possible to show that

$$\bigsqcup\{A \mid A \in \mathcal{A}(M)\} = \mathit{BT}(M)$$

where $\mathit{BT}(M)$ stands for the *Böhm tree* of M, a tree that represents the (possible infinite) normal form of M (see [10]).

2.2 Approximation result

In this subsection, the approximation theorem will be proved; the technique used differs slightly from that of [3]. For reasons of readability, in this subsection $\exists\, A \in \mathcal{A}(M)\,[B \vdash A{:}\sigma]$ will be abbreviated by $\mathit{Appr}\,(B, M, \sigma)$.

The following basic properties are needed further on.

Lemma 2.7 *i)* $\mathit{Appr}\,(B, x\overrightarrow{M_i}, \sigma{\to}\tau)\ \&\ \mathit{Appr}\,(B, N, \sigma) \Rightarrow \mathit{Appr}\,(B, x\overrightarrow{M_i}N, \tau)$.
ii) $\mathit{Appr}\,(B \cup \{z{:}\sigma\}, Mz, \tau)\ \&\ z \notin \mathit{fv}(M)\ \&\ \tau \in \mathcal{T}_s \Rightarrow \mathit{Appr}\,(B, M, \sigma{\to}\tau)$.
iii) $\mathit{Appr}\,(B, M[N/x]\overrightarrow{P}, \sigma) \Rightarrow \mathit{Appr}\,(B, (\lambda x.M)N\overrightarrow{P}, \sigma)$.

Proof: *i)* $A \in \mathcal{A}(x\overrightarrow{M_i})\ \&\ B \vdash A{:}\sigma{\to}\tau\ \&\ A' \in \mathcal{A}(N)\ \&\ B \vdash A'{:}\tau \Rightarrow$ (*2.4(i)* & $(\to E)$)
$AA' \in \mathcal{A}(x\overrightarrow{M_i}N)\ \&\ B \vdash AA'{:}\tau$.
ii) $A \in \mathcal{A}(Mz)\ \&\ B, z{:}\sigma \vdash A{:}\tau\ \&\ z \notin \mathit{fv}(M) \Rightarrow$ (*2.4(ii)*)
a) $A \equiv A'z\ \&\ z \notin \mathit{fv}(A')\ \&\ A' \in \mathcal{A}(M)\ \&\ B, z{:}\sigma \vdash A'z{:}\tau \Rightarrow$ (*1.7*)
$A' \in \mathcal{A}(M)\ \&\ B \vdash A'{:}\sigma{\to}\tau$.
b) $\lambda z.A \in \mathcal{A}(M)\ \&\ B, z{:}\sigma \vdash A{:}\tau \Rightarrow \lambda z.A \in \mathcal{A}(M)\ \&\ B \vdash \lambda z.A{:}\sigma{\to}\tau$.
iii) Since $M[N/x]\overrightarrow{P} =_\beta (\lambda x.M)N\overrightarrow{P}$, the result follows by Lemma *2.4(iii)*. ■

In order to prove, that for each term typeable in $\vdash$, an approximant with the same type can be found, a notion of computability is introduced.

Definition 2.8 (COMPUTABILITY PREDICATE) $\mathit{Comp}\,(B, M, \rho)$ is inductively defined by:
i) $\mathit{Comp}\,(B, M, \varphi) \iff \mathit{Appr}\,(B, M, \varphi)$.
ii) $\mathit{Comp}\,(B, M, \sigma{\to}\tau) \iff (\mathit{Comp}\,(B', N, \sigma) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, MN, \tau))$.
iii) $\mathit{Comp}\,(B, M, \cap_{\underline{n}}\sigma_i) \iff \forall\, i \in \underline{n}\ [\mathit{Comp}\,(B, M, \sigma_i)]$.

Notice that $\mathit{Comp}\,(B, M, \omega)$ holds as special case of part *(iii)*.

Lemma 2.9 *If* $\mathit{Comp}\,(B, M, \sigma)$, *and* $B'' \leq B$, *then* $\mathit{Comp}\,(B'', M, \sigma)$.

Proof: By induction on the definition of $\mathit{Comp}\,(\cdot)$.

$(\sigma = \varphi)$: $\mathit{Comp}\,(B, M, \varphi)\ \&\ B'' \leq B \Rightarrow \mathit{Appr}\,(B, M, \varphi) \Rightarrow$
$\mathit{Appr}\,(B'', M, \varphi) \Rightarrow \mathit{Comp}\,(B'', M, \varphi)$.

$(\sigma = \alpha{\to}\beta)$: $\mathit{Comp}\,(B, M, \alpha{\to}\beta) \Rightarrow$ (*2.8(ii)*)
$(\mathit{Comp}\,(B', Q, \alpha) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, MQ, \beta)) \Rightarrow$ (*IH*)
$(\mathit{Comp}\,(B', Q, \alpha) \Rightarrow \mathit{Comp}\,(\bigcap\{B'', B'\}, MQ, \beta)) \Rightarrow$ (*2.8(ii)*)
$\mathit{Comp}\,(B'', M, \alpha{\to}\beta)$.

$(\sigma = \cap_{\underline{n}}\sigma_i)$: $\mathit{Comp}\,(B, M, \cap_{\underline{n}}\sigma_i) \Rightarrow$ (*2.8(iii)*)
$\forall\, i \in \underline{n}\ [\mathit{Comp}\,(B, M, \sigma_i)] \Rightarrow$ (*IH*)
$\forall\, i \in \underline{n}\ [\mathit{Comp}\,(B'', M, \sigma_i)] \Rightarrow$ (*2.8(iii)*)
$\mathit{Comp}\,(B'', M, \cap_{\underline{n}}\sigma_i)$.

We will now show that the computability predicate is closed for $\leq$.

Lemma 2.10 *Take* σ *and* τ *such that* $\sigma \leq \tau$. *Then* $\mathit{Comp}\,(B, M, \sigma) \Rightarrow \mathit{Comp}\,(B, M, \tau)$.

Proof: By straightforward induction on the definition of $\leq$.

$(\cap_{\underline{n}}\sigma_i \leq \sigma_i\ (i \in \underline{n}))$: $\mathit{Comp}\,(B, M, \cap_{\underline{n}}\sigma_i) \Rightarrow$ (*2.8(iii)*) $\mathit{Comp}\,(B, M, \sigma_i)$.

$(\tau \leq \sigma_i\ (i \in \underline{n}) \Rightarrow \tau \leq \cap_{\underline{n}}\sigma_i)$: $\mathit{Comp}\,(B, M, \tau) \Rightarrow$ (*IH*)
$\mathit{Comp}\,(B, M, \sigma_i)\ (i \in \underline{n}) \Rightarrow$ (*2.8(iii)*) $\mathit{Comp}\,(B, M, \cap_{\underline{n}}\sigma_i)$.

$(\rho \leq \sigma\ \&\ \tau \leq \mu \Rightarrow \sigma{\to}\tau \leq \rho{\to}\mu)$: $\mathit{Comp}\,(B, M, \sigma{\to}\tau) \Rightarrow$ (*2.8(ii)*)
$(\mathit{Comp}\,(B', N, \sigma) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, MN, \tau)) \Rightarrow$ (*IH* 2×)
$(\mathit{Comp}\,(B', N, \rho) \Rightarrow \mathit{Comp}\,(B', N, \sigma) \Rightarrow$
$\mathit{Comp}\,(\bigcap\{B, B'\}, MN, \tau) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, MN, \mu)) \Rightarrow$
$(\mathit{Comp}\,(B', N, \rho) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, MN, \mu)) \Rightarrow$ (*2.8(ii)*)
$\mathit{Comp}\,(B, M, \rho{\to}\mu)$. ■

We will now show that the computability predicate is closed for β-expansion

Lemma 2.11 $\mathit{Comp}\,(B, M[N/x]\overrightarrow{P}, \sigma) \Rightarrow \mathit{Comp}\,(B, (\lambda x.M)N\overrightarrow{P}, \sigma)$.

Proof: By induction on the definition of $\mathit{Comp}\,(\cdot)$.

$(\sigma = \varphi)$: $\mathit{Comp}\,(B, M[N/x]\overrightarrow{P}, \varphi) \Rightarrow \mathit{Appr}\,(B, M[N/x]\overrightarrow{P}, \varphi) \Rightarrow$ (*2.7(iii)*)
$\mathit{Appr}\,(B, (\lambda x.M)N\overrightarrow{P}, \varphi) \Rightarrow \mathit{Comp}\,(B, (\lambda x.M)N\overrightarrow{P}, \varphi)$.

$(\sigma = \alpha{\to}\beta)$: $\mathit{Comp}\,(B, M[N/x]\overrightarrow{P}, \alpha{\to}\beta) \Rightarrow$ (*2.8(ii)*)
$(\mathit{Comp}\,(B', Q, \alpha) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, M[N/x]\overrightarrow{P}Q, \beta)) \Rightarrow$ (*IH*)
$(\mathit{Comp}\,(B', Q, \alpha) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, (\lambda x.M)N\overrightarrow{P}Q, \beta)) \Rightarrow$ (*2.8(ii)*)
$\mathit{Comp}\,(B, (\lambda x.M)N\overrightarrow{P}, \alpha{\to}\beta)$.

$(\sigma = \cap_{\underline{n}}\sigma_i)$: $\mathit{Comp}\,(B, M[N/x]\overrightarrow{P}, \cap_{\underline{n}}\sigma_i) \Rightarrow$ (*2.8(iii)*)
$\forall i \in \underline{n}\ [\mathit{Comp}\,(B, M[N/x]\overrightarrow{P}, \sigma_i)] \Rightarrow$ (*IH*)
$\forall i \in \underline{n}\ [\mathit{Comp}\,(B, (\lambda x.M)N\overrightarrow{P}, \sigma_i)] \Rightarrow$ (*2.8(iii)*)
$\mathit{Comp}\,(B, (\lambda x.M)N\overrightarrow{P}, \cap_{\underline{n}}\sigma_i)$.

The following theorem essentially shows that all term-variables are computable of any type, and that all terms computable of a certain type have an approximant with that same type.

Theorem 2.12 *i)* $\mathit{Appr}\,(B, x\overrightarrow{M_i}, \rho) \Rightarrow \mathit{Comp}\,(B, x\overrightarrow{M_i}, \rho)$.
ii) $\mathit{Comp}\,(B, M, \rho) \Rightarrow \mathit{Appr}\,(B, M, \rho)$.

Proof: Simultaneously by induction on the structure of types. The only interesting case is when $\rho = \sigma{\to}\tau$; when ρ is a type-variable, the result is immediate and when it is an intersection type, it is dealt with by induction.

i) $\mathit{Appr}\,(B, x\overrightarrow{M_i}, \sigma{\to}\tau) \Rightarrow$ (*IH (ii)*)
$(\mathit{Comp}\,(B', N, \sigma) \Rightarrow \mathit{Appr}\,(B, x\overrightarrow{M_i}, \sigma{\to}\tau)\ \&\ \mathit{Appr}\,(B', N, \sigma)) \Rightarrow$ (*2.7(i)*)
$(\mathit{Comp}\,(B', N, \sigma) \Rightarrow \mathit{Appr}\,(\bigcap\{B, B'\}, x\overrightarrow{M_i}N, \tau)) \Rightarrow$ (*IH (i)*)
$(\mathit{Comp}\,(B', N, \sigma) \Rightarrow \mathit{Comp}\,(\bigcap\{B, B'\}, x\overrightarrow{M_i}N, \tau)) \Rightarrow$ (*2.8(ii)*)
$\mathit{Comp}\,(B, x\overrightarrow{M_i}, \sigma{\to}\tau)$.

ii) $\mathit{Comp}\,(B, M, \sigma{\to}\tau)\ \&\ z \notin \mathit{fv}(M) \Rightarrow$ (*IH (i)*)
$\mathit{Comp}\,(B, M, \sigma{\to}\tau)\ \&\ \mathit{Comp}\,(\{z{:}\sigma\}, z, \sigma)\ \&\ z \notin \mathit{fv}(M) \Rightarrow$ (*2.8(ii)*)
$\mathit{Comp}\,(\bigcap\{B, \{z{:}\sigma\}\}, Mz, \tau)\ \&\ z \notin \mathit{fv}(M) \Rightarrow$ (*IH (ii)*)
$\mathit{Appr}\,(\bigcap\{B, \{z{:}\sigma\}\}, Mz, \sigma)\ \&\ z \notin \mathit{fv}(M) \Rightarrow$ (*2.7(ii)*)
$\mathit{Appr}\,(B, M, \sigma{\to}\tau)$. ■

Notice that, as a corollary of the first of these two results, we get that term-variables are computable for any type.

Corollary 2.13 *Comp* $(\{x{:}\sigma\}, x, \sigma)$, *for all* x, σ.

Theorem 2.14 *If* $\{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\} \vdash M{:}\sigma$, *and, for every* $i \in \underline{n}$, *Comp* (B_i, N_i, μ_i), *then Comp* $(\bigcap\{B_1, \ldots, B_n\}, M[\overrightarrow{N_i/x_i}], \sigma)$.

Proof: By induction on the structure of derivations; let $\{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\} = B_0$, and $B^0 = \bigcap\{B_1, \ldots, B_n\}$.

(*Ax*) : Then $M \equiv x_j$, for some $j \in \underline{n}$, $\mu_j \leq \sigma$, and $M[\overrightarrow{N_i/x_i}] \equiv x_j[\overrightarrow{N_i/x_i}] \equiv N_j$. From *Comp* (B_j, N_j, μ_j), by Lemma *2.10*, also *Comp* (B_j, N_j, σ), and, since $B^0 \leq B_j$, by Lemma 2.9, also *Comp* (B^0, N_j, σ).

($\rightarrow$*I*) : Then $M \equiv \lambda y.M'$, $\sigma = \rho{\rightarrow}\tau$, and $B_0, y{:}\rho \vdash M'{:}\tau$.

$\forall i \in \underline{n}\ [\textit{Comp}\,(B_i, N_i, \mu_i)]\ \&\ B_0, y{:}\rho \vdash M'{:}\tau\ \Rightarrow$ (*IH*)

(*Comp* $(B', N, \rho)\ \Rightarrow$ *Comp* $(\bigcap\{B^0, B'\}, M'[\overrightarrow{N_i/x_i}, N/y], \tau))\ \Rightarrow$ (*2.11*)

(*Comp* $(B', N, \rho)\ \Rightarrow$ *Comp* $(\bigcap\{B^0, B'\}, (\lambda y.M'[\overrightarrow{N_i/x_i}])N, \tau))\ \Rightarrow$ (*2.8(ii)*)

Comp $(B^0, (\lambda y.M')[\overrightarrow{N_i/x_i}], \rho{\rightarrow}\tau)$.

($\rightarrow$*E*) : Then $M \equiv M_1M_2$, $B_0 \vdash M_1{:}\rho{\rightarrow}\sigma$, and $B_0 \vdash M_2{:}\rho$.

$\forall i \in \underline{n}\ [\textit{Comp}\,(B_i, N_i, \mu_i)]\ \&\ B_0 \vdash M_1{:}\rho{\rightarrow}\sigma\ \&\ B_0 \vdash M_2{:}\rho\ \Rightarrow$ (*IH*)

Comp $(B^0, M_1[\overrightarrow{N_i/x_i}], \rho{\rightarrow}\sigma)$ & *Comp* $(B^0, M_2[\overrightarrow{N_i/x_i}], \rho)\ \Rightarrow$ (*2.8(ii)*)

Comp $(B^0, (M_1M_2)[\overrightarrow{N_i/x_i}], \sigma)$.

($\cap$*I*) : Straightforward by induction. ■

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

Theorem 2.15 (APPROXIMATION THEOREM) $B \vdash M{:}\sigma \iff \exists A \in \mathcal{A}(M)\ [B \vdash A{:}\sigma]$.

Proof: ($\Rightarrow$) : $B \vdash M{:}\sigma\ \Rightarrow$(*2.14* & *2.13*) *Comp* $(B, M, \sigma)\ \Rightarrow$(*2.12(ii)*) $\exists A \in \mathcal{A}(M)\ [B \vdash A{:}\sigma]$.

($\Leftarrow$) : Let $A \in \mathcal{A}(M)$ be such that $B \vdash A{:}\sigma$. Since $A \in \mathcal{A}(M)$, there is an M' such that $M' =_\beta M$ and $A \underset{\sim}{\sqsubset} M'$. Then, by Lemma *2.3*, $B \vdash M'{:}\sigma$ and, by Theorem *1.10*, also $B \vdash M{:}\sigma$. ■

2.3 Principal pairs and Semantics

For terms in $\mathcal{N}$, a notion of principal pair can be defined as follows:

Definition 2.16 (PRINCIPAL PAIR) *i*) Let $A \in \mathcal{N}$. $pp_{\mathcal{E}}(A)$, the *principal pair* of A, is defined by:

a) $pp_{\mathcal{E}}(\bot) = \langle , \omega\rangle$.

b) $pp_{\mathcal{E}}(x) = \langle\{x{:}\varphi\}, \varphi\rangle$.

c) If $A \neq \bot$, and $pp_{\mathcal{E}}(A) = \langle P, \pi\rangle$, then:

1) If x occurs free in A, and $x{:}\sigma \in P$, then $pp_{\mathcal{E}}(\lambda x.A) = \langle P\backslash x, \sigma{\rightarrow}\pi\rangle$.

2) Otherwise $pp_{\mathcal{E}}(\lambda x.A) = \langle P, \omega{\rightarrow}\pi\rangle$.

d) If for $i \in \underline{n}$, $pp_{\mathcal{E}}(A_i) = \langle P_i, \pi_i \rangle$ (disjoint in pairs), then

$$pp_{\mathcal{E}}(x\overrightarrow{A_i}) = \langle \bigcap\{P_1, \ldots, P_n, \{x{:}\pi_1 \to \cdots \to \pi_n \to \varphi\}\}, \varphi \rangle,$$

where φ is a type-variable that does not occur in $pp_{\mathcal{E}}(A_i)$, for $i \in \underline{n}$.

ii) $\mathcal{P} = \{\langle P, \pi \rangle \mid \exists\, A \in \mathcal{N}\, [pp_{\mathcal{E}}(A) = \langle P, \pi \rangle]\}$.

The definition is brought to arbitrary terms via:

Definition 2.17 ([3]) *i*) Let M be a term. Let $\Pi(M)$ be the set of all principal pairs for all approximants of M: $\Pi(M) = \{pp_{\mathcal{E}}(A) \mid A \in \mathcal{A}(M)\}$.

ii) $\Pi(M)$ is an ideal in $\mathcal{P}$, and therefore:

a) If $\Pi(M)$ is finite, then there exists a pair $\langle P, \pi \rangle = \bigsqcup \Pi(M)$, where $\langle P, \pi \rangle \in \mathcal{P}$. This pair is then called the principal pair of M.

b) If $\Pi(M)$ is infinite, $\bigsqcup \Pi(M)$ does not exist in $\mathcal{P}$. The principal pair of M is then the infinite set of pairs $\Pi(M)$.

That this gives indeed the *principal* pair for a term M is shown in [3].

Like in [13, 42, 2], it can be proved that there exists a precise relation between terms in $\mathcal{N}$ and principal pairs, both equipped with an appropriate ordering. Here, the relation $\preccurlyeq$ on pairs as given below is used.

Definition 2.18 ([3]) The relation on pairs $\preccurlyeq$ is defined by:

i) $\langle B, \sigma \rangle \preccurlyeq \langle \emptyset, \omega \rangle$.

ii) $\forall\, i \in \underline{n}\ (n \geq 2)\ [\langle B_i, \sigma_i \rangle \preccurlyeq \langle B'_i, \sigma'_i \rangle] \Rightarrow \langle \bigcap\{\underline{n}\}B_i, \cap_{\underline{n}}\sigma_i \rangle \preccurlyeq \langle \bigcap\{\underline{n}\}B'_i, \cap_{\underline{n}}\sigma'_i \rangle$.

iii) $\langle B \cup \{x{:}\rho\}, \mu \rangle \preccurlyeq \langle B' \cup \{x{:}\rho'\}, \mu' \rangle \Rightarrow \langle B, \rho \to \mu \rangle \preccurlyeq \langle B', \rho' \to \mu' \rangle$.

iv) $\forall\, i \in \underline{n}\ [\langle B_i, \sigma_i \rangle \preccurlyeq \langle B'_i, \sigma'_i \rangle] \Rightarrow \langle \bigcap\{B_1, \ldots, B_n, \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma\}\}, \sigma \rangle \preccurlyeq \langle \bigcap\{B'_1, \ldots, B'_n, \{x{:}\sigma'_1 \to \cdots \to \sigma'_n \to \sigma\}\}, \sigma \rangle$.

The following result links the approximant semantics to principal types.

Theorem 2.19 ([3]) *$\langle \mathcal{P}, \succcurlyeq \rangle$ is a meet semi-lattice isomorphic to $\langle \mathcal{N}, \underset{\sim}{\sqsubseteq} \rangle$.*

2.4 Normalization results

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

Lemma 2.20 *i) If $B \vdash A{:}\sigma$ and B, σ are ω-free, then A is $\bot$-free.*

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

Proof: By induction on A.

i) As before, only the part $\sigma \in \mathcal{T}_s$ is shown.

$(A \equiv \bot)$: Impossible, since $\bot$ is only typeable by ω.

$(A \equiv \lambda x.A')$: Then $\sigma = \alpha \to \beta$, and $B, x{:}\alpha \vdash A{:}\beta$. Since B, σ are ω-free, so are $B, x{:}\alpha$ and β, so, by induction, A' is $\bot$-free, so also $\lambda x.A'$ is $\bot$-free.

$(A \equiv x\overrightarrow{A_i})$: Then, by $(\to E)$ and (Ax), there are $\sigma_i\,(i \in \underline{n}), \tau_j\,(j \in \underline{n}), \tau$, such that $x{:}\tau_1 \to \cdots \to \tau_n \to \tau \in B$, for every $i \in \underline{n}$, $B \vdash A_i{:}\sigma_i$, and $\tau_1 \to \cdots \to \tau_n \to \tau \leq \sigma_1 \to \cdots \to \sigma_n \to \sigma$. So, especially, for every $i \in \underline{n}$, $\sigma_i \leq \tau_i$. By Theorem *1.8*, also for every $i \in \underline{n}$, $B \vdash A_i{:}\tau_i$. Since each τ_i occurs in B, all are ω-free, so by induction each A_i is $\bot$-free. Then also $x\overrightarrow{A_i}$ is $\bot$-free.

ii) *a*) $A \equiv \lambda x.A'$. By induction there are B, τ such that $B \vdash A'{:}\tau$ and B, τ are ω-free. If x does not occur in B, take an ω-free $\sigma \in \mathcal{T}_s$. Otherwise, there exist $x{:}\sigma \in B$, and σ is ω-free. In any case, $B\backslash x \vdash \lambda x.A'{:}\sigma \to \tau$, and $B\backslash x$ and $\sigma \to \tau$ are ω-free.

b) $A \equiv x\overrightarrow{A_i}$, with $(n \geq 0)$. By induction there are $B_i\,(i \in \underline{n})$ and $\sigma_i\,(i \in \underline{n})$ such that for every $i \in \underline{n}$, $B_i \vdash A_i{:}\sigma_i$, and B_i, σ_i are ω-free. Take σ strict, such that ω does not occur in σ, and $B = \bigcap\{B_1, \ldots, B_n, \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma\}\}$. Then $B \vdash x\overrightarrow{A_i}{:}\sigma$, and B and σ, are ω-free. ■

Now, as in [1] for the strict system, it is possible to prove that the type assignment system satisfies the main properties of the BCD-system.

Theorem 2.21 (NORMALISATION) *i)* $\exists B, \sigma\,[B \vdash M{:}\sigma \;\&\; B, \sigma$ *ω-free*$] \iff M$ *has a normal form.*

ii) $\exists B, \sigma \in \mathcal{T}_s\,[B \vdash M{:}\sigma] \iff M$ *has a head normal form.*

Proof: *i*) $(\Rightarrow)$: If $B \vdash M{:}\sigma$, then, by Theorem *2.15*, $\exists A \in \mathcal{A}(M)\,[B \vdash A{:}\sigma]$. Because of Lemma *2.20(i)*, this A is $\bot$-free. By Definition *2.1*, there exists $M' =_\beta M$ such that $A \sqsubseteq M'$. Since A is $\bot$-free, in fact $A \equiv M'$, so M' itself is in normal form, so, especially, M has a normal form.

$(\Leftarrow)$: If M' is the normal form of M, then it is a $\bot$-free approximate normal form. Then, by Lemma *2.20(ii)*, there are ω-free B, σ such that $B \vdash M'{:}\sigma$. Then, by Theorem *1.10*, $B \vdash M{:}\sigma$.

ii) $(\Rightarrow)$: If $B \vdash M{:}\sigma$, then, by Theorem *2.15*, $\exists A \in \mathcal{A}(M)\,[B \vdash A{:}\sigma]$. By Definition *2.1*, there exists $M' =_\beta M$ such that $A \sqsubseteq M$. Since $\sigma \in \mathcal{T}_s$, $A \not\equiv \bot$, so A is either $\lambda x.A_1$ or $x\overrightarrow{A_i}$, with $n \geq 0$. Since $A \sqsubseteq M'$, M' is either $\lambda x.M_1$, or $x\overrightarrow{M_i}$. Then M has a head-normal form.

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

1) $M' \equiv \lambda x.M_1$. Since M_1 is in head-normal form, by induction there are $B, \sigma \in \mathcal{T}_s$ such that $B \vdash M_1{:}\sigma$. If $x{:}\tau \in B$, then $B\backslash x \vdash \lambda x.M_1{:}\sigma \to \tau$, otherwise $B \vdash \lambda x.M_1{:}\omega \to \tau$.

2) $M' \equiv x\overrightarrow{M_i}, (n \geq 0)$. Take $\sigma \in \mathcal{T}_s$, then $\{x{:}\omega \to \cdots \to \omega \to \sigma\} \vdash x\overrightarrow{M_i}{:}\sigma$. ■

2.5 Strong normalisation

The other well-know result

$$B \vdash M{:}\sigma \textit{ without using } \omega \iff M \textit{ is strongly normalisable}$$

also holds, but needs a seperate proof in that it is not a consequence of the Approximation Theorem *2.15*. See [1] for a proof for this property for the BCD system that follows very much the structure of the proof of Theorem *2.15*, which could be applied directly here. Alternatively,

see [5] for a proof for the strict system where it is a direct consequence of the result that cut-elimination is strongly normalizable; this technique has not yet been extended to the system considered here.

We will now give an alternative proof. We shall prove that, when omega is removed from the system, every typeable term is strongly normalisable. This will be done using Tait-Girard's method.

In the sequel, we will accept the following without proof:

Fact 2.1 *i*) If $x\overrightarrow{M_i}$ and N are strongly normalizable, then so is $x\overrightarrow{M_i}N$.
ii) If $M[N/x]\overrightarrow{P}$ and N are strongly normalizable, then so is $(\lambda x.M)N\overrightarrow{P}$.

We use $\mathcal{SN}$ for the set of strongly normalisable terms.

Definition 2.22 We define the set $\mathcal{R}ed\lceil\rho\rfloor$ inductively over types by:

$$\begin{aligned}\mathcal{R}ed\lceil\varphi\rfloor &= \mathcal{SN}\\ \mathcal{R}ed\lceil\sigma{\to}\tau\rfloor &= \{M \mid \forall N\,[N \in \mathcal{R}ed\lceil\sigma\rfloor \Rightarrow MN \in \mathcal{R}ed\lceil\tau\rfloor]\}\\ \mathcal{R}ed\lceil\cap_{\underline{n}}\sigma_i\rfloor &= \bigcap_{1\le i\le n}\mathcal{R}ed\lceil\sigma_i\rfloor.\end{aligned}$$

We now show that reducibility implies strongly normalisability, and that all term variables are reducible. For the latter, we need to show that all typeable strongly normalisable terms that start with a term variable are reducible. The result then follows from the fact that each term variable is trivially strongly normalisable and that we can type any term variable with any type.

Lemma 2.23 *For all ρ,*
i) $\mathcal{R}ed\lceil\rho\rfloor \subseteq \mathcal{SN}$.
ii) $x\overrightarrow{N} \in \mathcal{SN} \Rightarrow x\overrightarrow{N} \in \mathcal{R}ed\lceil\rho\rfloor$.

Proof: By simultaneous induction on the structure of types, using Definition *2.22*. ■

i) (φ) : Immediate.
$(\sigma{\to}\tau)$: $M \in \mathcal{R}ed\lceil\sigma{\to}\tau\rfloor \Rightarrow$ (*IH(ii)*) $x \in \mathcal{R}ed\lceil\sigma\rfloor$ & $M \in \mathcal{R}ed\lceil\sigma{\to}\tau\rfloor \Rightarrow$ (*2.22*) $Mx \in \mathcal{R}ed\lceil\tau\rfloor \Rightarrow$ (*IH(i)*) $Mx \in \mathcal{SN} \Rightarrow M \in \mathcal{SN}$.
$(\cap_{\underline{n}}\sigma_i)$: $M \in \mathcal{R}ed\lceil\cap_{\underline{n}}\sigma_i\rfloor \Rightarrow$ (*2.22*) $M \in \mathcal{R}ed\lceil\sigma_i\rfloor \Rightarrow$ (*IH(ii)*) $M \in \mathcal{SN}$. ■

ii) (φ) : $x\overrightarrow{N} \in \mathcal{SN} \Rightarrow$ (*2.22*) $x\overrightarrow{N} \in \mathcal{R}ed\lceil\varphi\rfloor$.
$(\sigma{\to}\tau)$: $x\overrightarrow{N} \in \mathcal{SN}$ $\Rightarrow$ (*2.22* & *IH(i)*)
$P \in \mathcal{R}ed\lceil\sigma\rfloor \Rightarrow x\overrightarrow{N} \in \mathcal{SN}$ & $P \in \mathcal{SN}$ $\Rightarrow$ (*2.1(i)*)
$P \in \mathcal{R}ed\lceil\sigma\rfloor \Rightarrow x\overrightarrow{N}P \in \mathcal{SN}$ $\Rightarrow$ (*IH(ii)*)
$P \in \mathcal{R}ed\lceil\sigma\rfloor \Rightarrow x\overrightarrow{N}P \in \mathcal{R}ed\lceil\tau\rfloor$ $\Rightarrow$ (*2.22*)
$x\overrightarrow{N} \in \mathcal{R}ed\lceil\sigma{\to}\tau\rfloor$
$(\cap_{\underline{n}}\sigma_i)$: By Definition *2.22* and induction. ■

We will now show that the reducibility predicate is closed for $\le$.

Lemma 2.24 *Take σ and τ such that $\sigma \le \tau$. Then $\mathcal{R}ed\lceil\sigma\rfloor \subseteq \mathcal{R}ed\lceil\tau\rfloor$.*

Proof: By straightforward induction on the definition of $\le$.
$(\cap_{\underline{n}}\sigma_i \le \sigma_i\ (i \in \underline{n}))$: $\mathcal{R}ed\lceil\cap_{\underline{n}}\sigma_i\rfloor =$ (*2.22*) $\bigcap_{i\in\underline{n}}\mathcal{R}ed\lceil\sigma_i\rfloor \subseteq \mathcal{R}ed\lceil\sigma_i\rfloor$.

$(\tau \leq \sigma_i\ (i \in \underline{n}) \Rightarrow \tau \leq \cap_{\underline{n}}\sigma_i)$: $M \in \mathcal{Red}\lceil\tau\rfloor \Rightarrow$ (*IH*) $M \in \mathcal{Red}\lceil\sigma_i\rfloor\ (\forall i \in \underline{n}) \Rightarrow$
$M \in \bigcap_{i\in\underline{n}} \mathcal{Red}\lceil\sigma_i\rfloor \Rightarrow$ (*2.22*) $M \in \mathcal{Red}\lceil\cap_{\underline{n}}\sigma_i\rfloor$.

$(\rho \leq \sigma\ \&\ \tau \leq \mu \Rightarrow \sigma{\to}\tau \leq \rho{\to}\mu)$: $M \in \mathcal{Red}\lceil\sigma{\to}\tau\rfloor \Rightarrow$ (*2.22*)
$(N \in \mathcal{Red}\lceil\sigma\rfloor \Rightarrow MN \in \mathcal{Red}\lceil\tau\rfloor) \Rightarrow$ (*IH 2*$\times$)
$(N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow N \in \mathcal{Red}\lceil\sigma\rfloor \Rightarrow$
$MN \in \mathcal{Red}\lceil\tau\rfloor \Rightarrow MN \in \mathcal{Red}\lceil\mu\rfloor) \Rightarrow$
$(N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow MN \in \mathcal{Red}\lceil\mu\rfloor) \Rightarrow$ (*2.22*)
$M \in \mathcal{Red}\lceil\rho{\to}\mu\rfloor$. ■

We will now show that the reducibility predicate is closed for subject expansion.

Lemma 2.25 $M[N/x]\overrightarrow{P} \in \mathcal{Red}\lceil\sigma\rfloor\ \&\ N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow (\lambda x.M)N\overrightarrow{P} \in \mathcal{Red}\lceil\sigma\rfloor$.

Proof: By induction on the structure of types.

(φ) : $M[N/x]\overrightarrow{P} \in \mathcal{Red}\lceil\varphi\rfloor\ \&\ N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow$ (*2.22*)
$M[N/x]\overrightarrow{P} \in \mathcal{SN}\ \&\ N \in \mathcal{SN} \Rightarrow$ (*2.1 (ii)* & *(i)*)
$(\lambda x.M)N\overrightarrow{P} \in \mathcal{SN} \Rightarrow$ (*2.22*)
$(\lambda x.M)N\overrightarrow{P} \in \mathcal{Red}\lceil\varphi\rfloor$

$(\sigma{\to}\tau)$: $M[N/x]\overrightarrow{P} \in \mathcal{Red}\lceil\sigma{\to}\tau\rfloor\ \&\ N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow$ (*2.22*)
$Q \in \mathcal{Red}\lceil\sigma\rfloor \Rightarrow M[N/x]\overrightarrow{P}Q \in \mathcal{Red}\lceil\tau\rfloor\ \&\ N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow$ (*IH*)
$Q \in \mathcal{Red}\lceil\sigma\rfloor \Rightarrow (\lambda x.M)N\overrightarrow{P}Q \in \mathcal{Red}\lceil\tau\rfloor \Rightarrow$ (*2.22*)
$(\lambda x.M)N\overrightarrow{P} \in \mathcal{Red}\lceil\sigma{\to}\tau\rfloor$

$(\cap_{\underline{n}}\sigma_i)$: Directly by induction and Definition *2.22*. ■

We shall prove our strong normalisation result by showing that every typeable term is reducible. For this, we need to prove a stronger property: We will now show that if we replace term variables by reducible terms in a typeable term, we obtain a reducible term.

Theorem 2.26 *Let* $B = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$. *If* $B \vdash_{\mathcal{BCD}} M{:}\sigma$, *and, for* $i \in \underline{n}$, $N_i \in \mathcal{Red}\lceil\mu_i\rfloor$, *then* $M[\overrightarrow{N_i/x_i}] \in \mathcal{Red}\lceil\sigma\rfloor$.

Proof: By induction on the structure of derivations.

(*Ax*) : Then $M \equiv x_j$, for some $j \in \underline{n}$, $\mu_j \leq \sigma$, and $M[\overrightarrow{N_i/x_i}] \equiv x_j[\overrightarrow{N_i/x_i}] \equiv N_j$. From $N_j \in \mathcal{Red}\lceil\mu_j\rfloor$, by Lemma *2.24*, also $N_j \in \mathcal{Red}\lceil\sigma\rfloor$.

$(\to I)$: Then $M \equiv \lambda y.M'$, $\sigma = \rho{\to}\tau$, and $B, y{:}\rho \vdash_{\mathcal{BCD}} M'{:}\tau$.
$\forall i \in \underline{n}\ [N_i \in \mathcal{Red}\lceil\mu_i\rfloor]\ \&\ B, y{:}\rho \vdash_{\mathcal{BCD}} M'{:}\tau \Rightarrow$ (*IH*)
$N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow M'[\overrightarrow{N_i/x_i}, N/y] \in \mathcal{Red}\lceil\tau\rfloor \Rightarrow$ (*2.25*)
$N \in \mathcal{Red}\lceil\rho\rfloor \Rightarrow (\lambda y.M'[\overrightarrow{N_i/x_i}])N \in \mathcal{Red}\lceil\tau\rfloor \Rightarrow$ (*2.22*)
$(\lambda y.M')[\overrightarrow{N_i/x_i}] \in \mathcal{Red}\lceil\rho{\to}\tau\rfloor$.

$(\to E)$, $(\cap I)$: Straightforward by induction and Definition *2.22*. ■

Theorem 2.27 (STRONG NORMALISATION) *Any typeable term is strongly normalisable.*

Proof: By Lemma *2.23 (ii)*, all term variables are reducible of any type, so, by *2.26*, every typeable term is reducible. Strong normalisation then follows from Lemma *2.23 (i)*. ■

3 Semantics and completeness

3.1 Filter models

As in [11] and [1], a filter λ-model can be constructed.

Definition 3.1 (FILTERS) *i)* A subset d of $\mathcal{T}$ is a ***filter*** if and only if:
a) $\sigma_i \in d\ (i \in \underline{n}, n \geq 0) \Rightarrow \cap_{\underline{n}}\sigma_i \in d$.
b) $\tau \in d\ \&\ \tau \leq \sigma \Rightarrow \sigma \in d$.
ii) If V is a subset of $\mathcal{T}$, then $\uparrow V$ is the smallest filter that contains V, and $\uparrow\sigma = \uparrow\{\sigma\}$.
iii) $\mathcal{F}_S = \{d \subseteq \mathcal{T} \mid d \text{ is a filter}\}$. Application on $\mathcal{F}_S$ is defined by:

$$d \cdot e = \uparrow\{\tau \mid \exists\sigma \in e\ [\sigma{\rightarrow}\tau \in d]\}.$$

Notice that a filter is never empty; because of part *(i.a)*, for all d, $\omega \in d$. Notice that, as in [1], application must be forced to yield filters, since in each arrow type scheme $\sigma{\rightarrow}\tau \in \mathcal{T}$, τ is strict. $\langle\mathcal{F}_S, \subseteq\rangle$ is a cpo and henceforward it will be considered with the corresponding Scott topology.

For filters the following properties hold:

Lemma 3.2 *i)* $\sigma \in \uparrow\tau \iff \tau \leq \sigma$.
ii) $\sigma \in \uparrow\{\tau \mid B \vdash M{:}\tau\} \iff \sigma \in \{\tau \mid B \vdash M{:}\tau\}$. *(So* $\{\sigma \mid B \vdash M{:}\sigma\} \in \mathcal{F}_S$*.)*

Proof: Easy. ■

Definition 3.3 (DOMAIN CONSTRUCTORS) Define $F : \mathcal{F}_S{\rightarrow}[\mathcal{F}_S{\rightarrow}\mathcal{F}_S]$ and $G : [\mathcal{F}_S{\rightarrow}\mathcal{F}_S]{\rightarrow}\mathcal{F}_S$ by:
i) $F\ d\ e = d \cdot e$.
ii) $Gf = \uparrow\{\sigma{\rightarrow}\tau \mid \tau \in f(\uparrow\sigma)\}$.

It is easy to check that F and G are continuous.

Theorem 3.4 (FILTER MODEL) $\langle\mathcal{F}_S, \cdot, F, G\rangle$*, with* F *and* G *as defined in 3.3, is a* λ*-model.*

Proof: By [10].5.4.1 it is sufficient to prove that $F{\circ}G = Id_{[\mathcal{F}_S{\rightarrow}\mathcal{F}_S]}$.

$$\begin{array}{l} F{\circ}Gfd = F(Gf)d = \\ F(\uparrow\{\sigma{\rightarrow}\tau \mid \tau \in f(\uparrow\sigma)\})d = \\ \uparrow\{\mu \mid \exists\rho \in d\ [\rho{\rightarrow}\mu \in \uparrow\{\sigma{\rightarrow}\tau \mid \tau \in f(\uparrow\sigma)\}]\} = \ (3.2\,(i)) \\ \uparrow\{\mu \mid \exists\rho \in d\ [\mu \in f(\uparrow\rho)]\} = f(d). \end{array}$$ ■

Definition 3.5 (TERM INTERPRETATION) Let $\mathcal{M}$ be a lambda model, and ξ be a valuation of term variables in $\mathcal{M}$.
i) $[\![\cdot]\!]^{\mathcal{M}}_{\xi}$, the interpretation of terms in $\mathcal{M}$ via ξ is inductively defined by:
a) $[\![x]\!]^{\mathcal{M}}_{\xi} = \xi(x)$.
b) $[\![MN]\!]^{\mathcal{M}}_{\xi} = F[\![M]\!]^{\mathcal{M}}_{\xi}[\![N]\!]^{\mathcal{M}}_{\xi}$.
c) $[\![\lambda x.M]\!]^{\mathcal{M}}_{\xi} = G(\lambda\!\!\lambda\, d \in \mathcal{M}.[\![M]\!]^{\mathcal{M}}_{\xi(d/x)})$.
ii) $B_{\xi} = \{x{:}\sigma \mid \sigma \in \xi(x)\}$.

Since $\mathcal{F}_S$ is the model studied here, $[\![\cdot]\!]_\xi$ stands for $[\![\cdot]\!]_\xi^{\mathcal{F}_S}$. Notice that B_ξ is not really a basis, since it can contain infinitely many statements with subject x; however, for all its design and purposes, it can be regarded as one.

Theorem 3.6 *For all M, ξ: $[\![M]\!]_\xi = \{\sigma \mid B_\xi \vdash M{:}\sigma\}$.*

Proof: By induction on the structure of lambda terms.

i) $[\![x]\!]_\xi = \xi(x)$. If $\sigma \in \xi(x)$, then certainly $B_\xi \vdash x{:}\sigma$. Assume $B_\xi \vdash x{:}\sigma$: if $x{:}\rho \in B_\xi$, then $\rho \leq \sigma$, so $\sigma \in \uparrow\rho$. Since $\rho \in \xi(x)$, also $\uparrow\rho \subseteq \xi(x)$, so $\sigma \in \xi(x)$.

ii)
$$
\begin{array}{ll}
[\![MN]\!]_\xi = & \\
F\,[\![M]\!]_\xi\,[\![N]\!]_\xi = & \\
[\![M]\!]_\xi \cdot [\![N]\!]_\xi = & (\textit{IH}) \\
\{\rho \mid B_\xi \vdash M{:}\rho\} \cdot \{\rho \mid B_\xi \vdash N{:}\rho\} = & (\textit{3.1(iii)}) \\
\uparrow\{\tau \mid \exists\sigma \in \{\rho \mid B_\xi \vdash N{:}\rho\}\,[\sigma{\to}\tau \in \{\rho \mid B_\xi \vdash M{:}\rho\}]\} = & \\
\uparrow\{\tau \mid \exists\sigma\,[B_\xi \vdash N{:}\sigma \;\&\; B_\xi \vdash M{:}\sigma{\to}\tau]\} = & (\to\!\textit{E}) \\
\uparrow\{\tau \mid B_\xi \vdash MN{:}\tau\} = & (\textit{3.2(ii)}) \\
\{\tau \mid B_\xi \vdash MN{:}\tau\} &
\end{array}
$$

iii)
$$
\begin{array}{ll}
[\![\lambda x.M]\!]_\xi = & \\
G(\lambda\!\!\lambda\, d \in \mathcal{F}_S.[\![M]\!]_{\xi(d/x)}) = & (\textit{IH}) \\
G(\lambda\!\!\lambda\, d \in \mathcal{F}_S.\{\rho \mid B_{\xi(d/x)} \vdash M{:}\rho\}) = & \\
\uparrow\{\sigma{\to}\tau \mid \tau \in (\lambda\!\!\lambda\, d \in \mathcal{F}_S.\{\rho \mid B_{\xi(d/x)} \vdash M{:}\rho\})(\uparrow\sigma)\} = & \\
\uparrow\{\sigma{\to}\tau \mid \tau \in \{\rho \mid B_{\xi(\uparrow\sigma/x)} \vdash M{:}\rho\}\} = & \\
\uparrow\{\sigma{\to}\tau \mid B_{\xi(\uparrow\sigma/x)} \vdash M{:}\tau\} = & (B'_\xi = B_\xi\backslash x) \\
\uparrow\{\sigma{\to}\tau \mid B'_\xi \cup \{x{:}\mu \mid \mu \in \uparrow\sigma\} \vdash M{:}\tau\} = & (\textit{3.2(i)} \;\&\; \textit{1.5(vi)}) \\
\uparrow\{\sigma{\to}\tau \mid B'_\xi \cup \{x{:}\sigma\} \vdash M{:}\tau\} = & (\to\!\textit{I}) \\
\uparrow\{\sigma{\to}\tau \mid B'_\xi \vdash \lambda x.M{:}\sigma{\to}\tau\} = & (\textit{1.5(v)}) \\
\uparrow\{\sigma{\to}\tau \mid B_\xi \vdash \lambda x.M{:}\sigma{\to}\tau\} = & ((\to\!\textit{I}) \;\&\; \textit{3.2(ii)}) \\
\{\rho \mid B_\xi \vdash \lambda x.M{:}\rho\}. & \blacksquare
\end{array}
$$

3.2 Soundness and completeness of type assignment

The main result of [11] is the proof for completeness of type assignment.

In constructing a complete system, the semantics of types plays a crucial role.

Definition 3.7 (TYPE INTERPRETATION) Let $\langle \mathcal{D}, \cdot, \varepsilon \rangle$ be a continuous λ-model. A mapping $\upsilon : \mathcal{T} \to \wp(\mathcal{D}) = \{X \mid X \subseteq \mathcal{D}\}$ is an *type interpretation* if and only if:

i) $\upsilon(\sigma{\to}\tau) = \{d \mid \forall e \in \upsilon(\sigma)\,[d \cdot e \in \upsilon(\tau)]\}$.

ii) $\upsilon(\sigma{\cap}\tau) = \upsilon(\sigma) \cap \upsilon(\tau)$.

Lemma 3.8 Let υ be a type interpretation. Then $\sigma \leq \tau$ implies $\upsilon(\sigma) \subseteq \upsilon(\tau)$.

Proof: Easy.

This notion of type interpretation leads, naturally, to the definition for semantic satisfiability.

Definition 3.9 (SATISFIABILITY) *i*) Let $\mathcal{M} = \langle \mathcal{D}, \cdot, [\![\ ]\!] \rangle$ be a λ-model and ξ a valuation of term-variables in $\mathcal{D}$. Then $[\![M]\!]^{\mathcal{M}}_{\xi} \in \mathcal{D}$ is the interpretation of M in $\mathcal{M}$ via ξ.

ii) Define $\models$ by (where $\mathcal{M}$ is a λ-model, ξ a valuation and υ a type interpretation);

a) $\mathcal{M}, \xi, \upsilon \models M{:}\sigma \iff [\![M]\!]^{\mathcal{M}}_{\xi} \in \upsilon(\sigma)$.

b) $\mathcal{M}, \xi, \upsilon \models B \iff \mathcal{M}, \xi, \upsilon \models x{:}\sigma$ for every $x{:}\sigma \in B$.

c) $B \models M{:}\sigma \iff \forall \mathcal{M}, \xi, \upsilon\, [\mathcal{M}, \xi, \upsilon \models B \Rightarrow \mathcal{M}, \xi, \upsilon \models M{:}\sigma]$.

Since no confusion is possible, the superscript on $[\![\ ]\!]$ is omitted.

Theorem 3.10 (SOUNDNESS) $B \vdash M{:}\sigma \Rightarrow B \models M{:}\sigma$.

Proof: By Definition *3.9(ii.c)*, for all $\mathcal{M}, \xi, \upsilon$, if $\mathcal{M}, \xi, \upsilon \models B$ then $\mathcal{M}, \xi, \upsilon \models M{:}\sigma$. This then means that, if $\mathcal{M}, \xi, \upsilon \models x{:}\rho$ for every $x{:}\rho \in B$, then $\mathcal{M}, \xi, \upsilon \models M{:}\sigma$, so, to show:

$$\text{if } [\![x]\!]^{\mathcal{M}}_{\xi} \in \upsilon(\rho) \text{ for every } x{:}\rho \in B, \text{ then } [\![M]\!]^{\mathcal{M}}_{\xi} \in \upsilon(\sigma).$$

We prove the property for the model $\mathcal{F}_S$, by induction on the structure of derivations.

(*Ax*) : Then $B \vdash x{:}\sigma$, so there exists $x{:}\rho \in B$ such that $\rho \leq \sigma$. Assume $[\![x]\!]_{\xi} \in \upsilon(\rho)$, then, by Lemma *3.8*, $[\![x]\!]_{\xi} \in \upsilon(\sigma)$.

($\rightarrow$*I*) : Then $B \vdash \lambda y.M'{:}\alpha{\rightarrow}\beta$, and also $B, y{:}\alpha \vdash M'{:}\beta$. Let $e \in \upsilon(\alpha)$, and $\xi' = \xi(e/y)$.

Then: $\forall x{:}\tau \in B, y{:}\alpha\, [[\![x]\!]_{\xi'} \in \upsilon(\tau)] \Rightarrow$ (*IH*)

$[\![M']\!]_{\xi'} \in \upsilon(\beta) \Rightarrow$ (*3.6*)

$\{\delta \mid B_{\xi'} \vdash M'{:}\delta\} \in \upsilon(\beta) \Rightarrow$ ($\rightarrow$*I*)

$\uparrow\{\delta \mid \exists \gamma \in e\, [B_{\xi} \vdash \lambda y.M'{:}\gamma{\rightarrow}\delta]\} \in \upsilon(\beta) \Rightarrow$

$\uparrow\{\delta \mid \exists \gamma \in e\, [\gamma{\rightarrow}\delta \in \{\rho \mid B_{\xi} \vdash \lambda y.M'{:}\rho\}]\} \in \upsilon(\beta) \Rightarrow$ (*3.1(iii)*)

$\{\rho \mid B_{\xi} \vdash \lambda y.M'{:}\rho\} \cdot e \in \upsilon(\beta)$.

So, for all $e \in \upsilon(\alpha)$, we have shown that $\{\rho \mid B_{\xi} \vdash \lambda y.M'{:}\rho\} \cdot e \in \upsilon(\beta)$, so, by Definition *3.7*, we get $\{\rho \mid B_{\xi} \vdash \lambda y.M'{:}\rho\} \in \upsilon(\alpha{\rightarrow}\beta)$.

($\rightarrow$*E*) : Then $M \equiv PQ$, and there exists μ such that $B \vdash P{:}\mu{\rightarrow}\sigma$ and $B \vdash Q{:}\mu$.

Then: $\forall x{:}\tau \in B, y{:}\alpha\, [[\![x]\!]_{\xi} \in \upsilon(\tau)] \Rightarrow$ (*IH*)

$[\![P]\!]_{\xi} \in \upsilon(\mu{\rightarrow}\sigma)\ \&\ [\![Q]\!]_{\xi} \in \upsilon(\mu) \Rightarrow$ (*3.6*)

$\{\rho \mid B_{\xi} \vdash P{:}\rho\} \in \upsilon(\mu{\rightarrow}\rho)\ \&\ \{\rho \mid B_{\xi} \vdash Q{:}\rho\} \in \upsilon(\mu) \Rightarrow$ ($\rightarrow$*I*)

$\{\rho \mid B_{\xi} \vdash P{:}\rho\} \in \{d \mid \forall e \in \upsilon(\mu)\, [d \cdot e \in \upsilon(\tau)\}\ \&$

$\{\rho \mid B_{\xi} \vdash Q{:}\rho\} \in \upsilon(\mu) \Rightarrow$

$\{\rho \mid B_{\xi} \vdash P{:}\rho\} \cdot \{\rho \mid B_{\xi} \vdash Q{:}\rho\} \in \upsilon(\tau) \Rightarrow$ (*3.1(iii)*)

$\uparrow\{\beta \mid \exists \alpha \in \{\rho \mid B_{\xi} \vdash Q{:}\rho\}\, [\alpha{\rightarrow}\beta \in \{\rho \mid B_{\xi} \vdash P{:}\rho\}]\} \in \upsilon(\tau) \Rightarrow$

$\uparrow\{\beta \mid \exists \alpha\, [B_{\xi} \vdash Q{:}\alpha\ \&\ B_{\xi} \vdash P{:}\alpha{\rightarrow}\beta]\} \in \upsilon(\tau) \Rightarrow$ ($\rightarrow$*E*)

$\uparrow\{\beta \mid B_{\xi} \vdash PQ{:}\beta\} \in \upsilon(\tau) \Rightarrow$

$\{\beta \mid B_{\xi} \vdash PQ{:}\beta\} \in \upsilon(\tau)$.

($\cap$*I*) : Then $\sigma = \cap_{\underline{n}}\sigma_i$, and, for $i \in \underline{n}$, $B \vdash M{:}\sigma_i$.

Then: $\forall x{:}\tau \in B, y{:}\alpha\, [[\![x]\!]_{\xi} \in \upsilon(\tau)] \Rightarrow$ (*IH*)

$\forall i \in \underline{n}\, [\{\rho \mid B_{\xi} \vdash M{:}\rho\} \in \upsilon(\sigma_i)] \Rightarrow$

$\{\rho \mid B_{\xi} \vdash M{:}\rho\} \in \upsilon(\sigma_1) \cap \cdots \cap \upsilon(\sigma_n)] \Rightarrow$ (*3.7*)

$\{\rho \mid B_{\xi} \vdash M{:}\rho\} \in \upsilon(\cap_{\underline{n}}\sigma_i)$.

■

The method followed in [11] for the proof of completeness of type assignment is to define

a type interpretation υ that satisfies: for all types σ, $\upsilon(\sigma) = \{d \in \mathcal{F}_S \mid \sigma \in d\}$. The approach taken here is to define a function, and to show that it is a type interpretation.

Theorem 3.11 *The map ν_0 defined by: $\nu_0(\sigma) = \{d \in \mathcal{F}_S \mid \sigma \in d\}$ is a type interpretation.*

Proof: It is sufficient to check the conditions of Definition *3.7*:

$(\nu_0(\sigma{\to}\tau) = \{d \mid \forall e \in \nu_0(\sigma)\,[d \cdot e \in \nu_0(\tau)]\})$:

$\forall e\,[e \in \nu_0(\sigma) \Rightarrow d \cdot e \in \nu_0(\tau)] \iff$ (*3.1(iii)*)

$\forall e\,[e \in \nu_0(\sigma) \Rightarrow \uparrow\{\beta \mid \exists \alpha \in e\,[\alpha{\to}\beta \in d]\} \in \nu_0(\tau)] \iff$

$\forall e\,[\sigma \in e \Rightarrow \tau \in \uparrow\{\beta \mid \exists \alpha \in e\,[\alpha{\to}\beta \in d]\}] \iff$ ($\tau \in \mathcal{T}_s$)

$\forall e\,[\sigma \in e \Rightarrow \exists \alpha \in e\,[\alpha{\to}\tau \in d]] \iff$ ($\Rightarrow$: take $e = \uparrow\sigma$)

$\sigma{\to}\tau \in d \iff d \in \nu_0(\sigma{\to}\tau)$

$(\nu_0(\sigma{\cap}\tau) = \nu_0(\sigma) \cap \nu_0(\tau))$: Easy. ■

Lemma 3.12 *i)* $B \vdash M{:}\sigma \iff B_{\xi_B} \vdash M{:}\sigma$.

ii) $\mathcal{F}_S, \xi_B, \nu_0 \models B$.

Proof: *i*) Because for every x, $\xi_B(x)$ is a filter.

ii) $x{:}\sigma \in B \Rightarrow$ (*(i)*)$\sigma \in \{\tau \mid B_{\xi_B} \vdash x{:}\tau\} \Rightarrow \sigma \in [\![x]\!]_{\xi_B}$.

So $[\![x]\!]_{\xi_B} \in \{d \in \mathcal{F}_S \mid \sigma \in d\} = \nu_0(\sigma)$. ■

Since the interpretation of terms by their derivable types gives a λ-model, the following corollary is immediate and an alternative proof for Theorem *1.10*.

Corollary 3.13 *If $M =_\beta N$ and $B \vdash M{:}\sigma$, then $B \vdash N{:}\sigma$.*

Proof: Since $\mathcal{F}_S$ is a λ-model, if $M =_\beta N$, then $[\![M]\!]_\xi = [\![N]\!]_\xi$, for any ξ, and, by Lemma *3.12(i)*, $\{\sigma \mid B \vdash M{:}\sigma\} = \{\sigma \mid B \vdash N{:}\sigma\}$. ■

Theorem 3.14 (COMPLETENESS) *Let $\sigma \in \mathcal{T}$, then $B \models M{:}\sigma \Rightarrow B \vdash M{:}\sigma$.*

Proof:

$B \models M{:}\sigma \Rightarrow$ (*3.9(ii.c)*, *3.12(ii)* & *3.11*)

$\mathcal{F}_S, \xi_B, \nu_0 \models M{:}\sigma \Rightarrow$ (*3.9(i)*)

$[\![M]\!]_{\xi_B} \in \nu_0(\sigma) \Rightarrow$

$\sigma \in [\![M]\!]_{\xi_B} \Rightarrow$ (*3.6*)

$B_{\xi_B} \vdash M{:}\sigma \Rightarrow$ (*3.12(i)*)

$B \vdash M{:}\sigma$. ■

4 Combinator Systems

In this section, we will give a detailed presentation of Combinator Systems (CS). CS will be defined as a special kind of applicative TRS [32], with the restriction that formal parameters of function symbols are not allowed to have structure, and right-hand sides of term rewriting rules are constructed of term-variables only. We have chosen this kind of presentation in view of a future extension of the results to full TRS, in the spirit of [8]. Notice that our treatment differs from, for example, that of [21], where only combinatory complete CS are considered.

Definition 4.1 (COMBINATOR TERMS) *i*) An *alphabet* or *signature* $\Sigma = (\mathcal{C}, \mathcal{X})$ consists of a countable infinite set $\mathcal{X}$ of variables ranged over by $x, y, z, \ldots$, a non-empty set $\mathcal{C} =$

$\{\mathsf{D}, \mathsf{Z}, \ldots\}$ of *combinators*, ranged over by $C, D, E, \ldots$, each equipped with an arity greater than 0, and the binary function symbol *Ap* (application).

ii) The set $\mathcal{T}(\mathcal{C}, \mathcal{V})$ of *terms*, ranged over by t, is defined by:

$$t ::= x \mid C \mid \mathit{Ap}\,(t_1, t_2)$$

As usual, we will write $(t_1\, t_2)$ instead of $\mathit{Ap}\,(t_1, t_2)$, and left-most, outermost brackets will be omitted, so $t_1\, t_2\, (t_3 t_4)$ stands for $\mathit{Ap}\,(\mathit{Ap}\,(t_1, t_2), \mathit{Ap}\,(t_3, t_4))$.

The following is the usual notion of term-substitution formulated for combinator systems.

Definition 4.2 (TERM-SUBSTITUTIONS) A *term-substitution* R is a map from terms to terms, determined by its restriction to a finite set of variables, satisfying $\mathrm{R}\,(t_1\, t_2) = \mathrm{R}\,(t_1)\,\mathrm{R}\,(t_2)$. We will write t^{R} instead of $\mathrm{R}\,(t)$. If R maps x_i to u_i, for $i \in \underline{n}$, we write $\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$ for R, and write $t^{\vec{u}}$ for t^{R}.

Combinator Systems, and the notion of rewriting on combinator terms, are defined by the following:

Definition 4.3 (COMBINATOR SYSTEMS) *i*) A *combinator rule* on $\Sigma = (\mathcal{C}, \mathcal{X})$ is a pair (l, r) of terms in $\mathcal{T}(\mathcal{C}, \mathcal{V})$, such that:

a) There are C and distinct $x_1, \ldots, x_n$, such that $l = C x_1 \cdots x_n$, where $n = \mathit{arity}(C)$.

b) The variables occurring in r are contained in l, and r contains no symbols from $\mathcal{C}$.

ii) A *Combinator System* (CS) is a pair of an alphabet Σ and a set **R** of combinator rules on $\Sigma = (\mathcal{C}, \mathcal{X})$, such that there is *exactly one* rule in **R** for each combinator $C \in \mathcal{C}$. This rule (l, r) is called *the combinator rule for* C; we will use the symbol C also as name for this rule and write $l \to_C r$.

iii) A combinator rule $l \to_C r$ determines a set of *reductions* $l^{\mathrm{R}} \to_C r^{\mathrm{R}}$ for all term-substitutions R. The left-hand side l^{R} is called a *redex*; it may be replaced by its '*contractum*' r^{R} inside any context $\mathrm{C}[\ ]$; this gives rise to *reduction steps*: $\mathrm{C}[\, l^{\mathrm{R}}\,] \to_C \mathrm{C}[\, r^{\mathrm{R}}\,]$.

iv) We will write $t \to_{\mathbf{R}} t'$ if there is a rule $l \to_C r$ in **R** such that $t \to_C t'$, and call $\to_{\mathbf{R}}$ the *one-step rewrite relation* generated by **R**, and $\to_{\mathbf{R}}^+$ (respectively $\to_{\mathbf{R}}^*$) the transitive (respectively reflexive and transitive) closure of $\to_{\mathbf{R}}$ (the index **R** will be omitted when it is clear from the context). If $t_0 \to^+ t_n$, then t_n is a *reduct* of t_0.

Example 4.4 (COMBINATORY LOGIC) The standard example of a CS is Combinatory Logic (CL) – defined by Curry independently of LC [16] – that is, in our notation, formulated as follows: $\mathrm{CL} = (((\mathsf{S}, \mathsf{K}, \mathsf{I}), \mathcal{X}), \mathbf{R})$, where **R** contains the rules

$$\begin{aligned} \mathsf{S}\, x\, y\, z &\to x\, z\, (y\, z) \\ \mathsf{K}\, x\, y &\to x \\ \mathsf{I}\, x &\to x \end{aligned}$$

The last rule was not part of the original definition, but is nowadays normally added.

We will assume that no two combinators have the same interpretation in LC (see Definition *4.7*), so a CS like

$$\begin{aligned} \mathsf{I}\, x &\to x \\ \mathsf{J}\, x &\to x \end{aligned}$$

is excluded, since it would give an immediate counter example against any full-abstraction result with respect to the filter semantics (see Section *7*).

This notion of reduction on combinator terms as in Definition *4.3* is also known as *weak reduction* and satisfies the Church-Rosser Property (see [10]).

Proposition 4.5 (CHURCH-ROSSER) *If* $t \to^* u$ *and* $t \to^* v$*, then there exists* w *such that* $u \to^* w$ *and* $v \to^* w$. ■

We now define (head-)normal forms, (head-)normalizability, strongly normalizability, and unsolvable terms.

Definition 4.6 ((HEAD-)NORMAL FORMS) Let $((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS.

i) A term is in *normal form* with respect to **R** if it is irreducible.

ii) A term t is in *head-normal form* with respect to **R** if either
  a) there are a variable x and terms $t_1,\ldots,t_n$ $(n \geq 0)$ such that $t \equiv x\,t_1\cdots t_n$, or
  b) there are a combinator $C \in \mathcal{C}$ and terms $t_1,\ldots,t_n \in \mathcal{T}(\mathcal{C},\mathcal{V})$ such that $t \equiv C\,t_1\cdots t_n$, and $n < \mathit{arity}(C)$.

iii) A term is *(head-)normalizable* if it can be reduced to a term in (head-)normal form. A rewrite system is *strongly normalizing* (or terminating) if all rewrite sequences are finite; it is *(head-)normalizing* if every term is (head-)normalizable.

iv) A term is called *unsolvable* if it has no head-normal form.

4.1 CS versus LC

We now focus on the relation between reduction in CS and in LC.

Definition 4.7 Let $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS. $\langle\ \rangle^{\mathbf{C}}_{\lambda} : \mathcal{T}(\mathcal{C},\mathcal{V}) \to \Lambda$, the interpretation of combinator terms over **C** in LC, is defined by:

$$\begin{array}{rcll} \langle x\rangle^{\mathbf{C}}_{\lambda} & = & x & \text{for all } x \in \mathcal{X} \\ \langle t_1\,t_2\rangle^{\mathbf{C}}_{\lambda} & = & \langle t_1\rangle^{\mathbf{C}}_{\lambda}\langle t_2\rangle^{\mathbf{C}}_{\lambda} & \\ \langle C\rangle^{\mathbf{C}}_{\lambda} & = & \lambda x_1\cdots x_n.\langle r\rangle^{\mathbf{C}}_{\lambda} & \text{where } C\,x_1\cdots x_n \to r \in \mathbf{R} \end{array}$$

Notice that, since we assume the set of term variables for CS and LC to be the same, as well as the two notions of application on terms, $\langle r\rangle^{\mathbf{C}}_{\lambda} = r$ for every r that is the right-hand side of a combinator rule.

The interpretation in LC of a CS, $\langle\ \rangle^{\mathbf{C}}_{\lambda}$, respects reduction:

Proposition 4.8 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS*, then, for all* $t,t' \in \mathcal{T}(\mathcal{C},\mathcal{V})$*: if* $t \to^* t'$*, then* $\langle t\rangle^{\mathbf{C}}_{\lambda} \twoheadrightarrow_{\beta} \langle t'\rangle^{\mathbf{C}}_{\lambda}$.

Proof: By easy induction on the definition of $\to^*$. ■

In general, the length of the reduction sequence increases significantly.

Only for particular CS it is possible to also define an interpretation of LC; the standard example is that of CL (see also [18, 10, 21]; in [21] also other combinatory complete CS are discussed).

Definition 4.9 The mapping $[\![\ ]\!]_{\mathrm{CL}} : \Lambda \to \mathcal{T}_{\mathrm{CL}}$ is defined by:

$$\begin{aligned} [\![x]\!]_{\mathrm{CL}} &= x \\ [\![\lambda x.M]\!]_{\mathrm{CL}} &= \lambda^* x.[\![M]\!]_{\mathrm{CL}} \\ [\![MN]\!]_{\mathrm{CL}} &= [\![M]\!]_{\mathrm{CL}}\,[\![N]\!]_{\mathrm{CL}} \end{aligned}$$

where $\lambda^* x.t$, with $t \in \mathcal{T}_{\mathrm{CL}}$, is defined by induction on the structure of t:

$$\begin{aligned} \lambda^* x.x &= \mathsf{I} \\ \lambda^* x.t &= \mathsf{K}t \qquad\qquad \text{if } x \text{ not in } t \\ \lambda^* x.t_1 t_2 &= \mathsf{S}(\lambda^* x.t_1)(\lambda^* x.t_2) \end{aligned}$$

For the interpretations defined above the following property holds:

Exercise 4.1 ([10]) *i)* $(\lambda^* x.t)\,v \to t^{x \mapsto v}$.
ii) $\langle \lambda^* x.t \rangle_\lambda^{\mathsf{C}} \twoheadrightarrow_\beta \lambda x.\langle t \rangle_\lambda^{\mathsf{C}}$
iii) $\langle [\![M]\!]_{\mathrm{CL}} \rangle_\lambda^{\mathrm{CL}} \twoheadrightarrow_\beta M$.
iv) If $t \to u$ in CL, then $\langle t \rangle_\lambda^{\mathrm{CL}} \twoheadrightarrow_\beta \langle u \rangle_\lambda^{\mathrm{CL}}$. ■

For example,

$$[\![\lambda xy.x]\!]_{\mathrm{CL}} = \lambda^* x.[\![\lambda y.x]\!]_{\mathrm{CL}} = \lambda^* x.(\lambda^* y.x) = \lambda^* x.(\mathsf{K}x) = \mathsf{S}(\lambda^* x.\mathsf{K})(\lambda^* x.x) = \mathsf{S}(\mathsf{KK})\mathsf{I}$$

and

$$\langle [\![\lambda xy.x]\!]_{\mathrm{CL}} \rangle_\lambda^{\mathrm{CL}} = \langle \mathsf{S}(\mathsf{KK})\mathsf{I} \rangle_\lambda^{\mathrm{CL}} = (\lambda xyz.xz(yz))((\lambda xy.x)\lambda xy.x)\lambda x.x \twoheadrightarrow_\beta \lambda xy.x.$$

There exists no converse of this property; moreover, the mapping $\langle\ \rangle_\lambda^{\mathrm{CL}}$ does not preserve normal forms or reductions:

Example 4.10 ([10]) *i)* SK is a normal form, but $\langle \mathsf{SK} \rangle_\lambda^{\mathrm{CL}} \twoheadrightarrow_\beta \lambda xy.y$,
ii) $t = \mathsf{S}(\mathsf{K}(\mathsf{SII}))(\mathsf{K}(\mathsf{SII}))$ is a normal form, but $\langle t \rangle_\lambda^{\mathrm{CL}} \twoheadrightarrow_\beta \lambda c.(\lambda x.xx)(\lambda x.xx)$, which does not have a β-normal form,
iii) $t = \mathsf{SK}(\mathsf{SII}(\mathsf{SII}))$ has no normal form, while $\langle t \rangle_\lambda^{\mathrm{CL}} \twoheadrightarrow_\beta \lambda x.x$.

We will show in Section *6.1* that the combinatorial equivalent of a well-known result for intersection type assignment in the LC, i.e. the property that normalising terms can be typed with a type not containing ω, no longer holds. Take for example the CS

$$\begin{aligned} \mathsf{Z}xy &\to y \\ \mathsf{D}x &\to xx \end{aligned}$$

then $\mathsf{Z}(\mathsf{DD})$ is typeable with a type not containing ω (see Example *6.17*). Notice that, since $\mathsf{DD} \to \mathsf{DD} \to \cdots$, the term $\mathsf{Z}(\mathsf{DD})$ has no normal form.

As these examples show, normalization results of LC do not transfer easily to CS. Here, we will study the normalization properties of CS directly in the CS framework.

5 Type assignment for CS

In this section, we will develop a notion of type assignment on CS that uses intersection types. It is inspired by similar definitions presented in [21] and [8]. As in [21], we will assume that,

for every combinator C, there is a basic type from which all types needed for an occurrence of C in a term can be obtained. The extension with respect to [21] is that we will not limit ourselves to basic types that are the principal type of the corresponding lambda term (see [42, 2]). The differences with [8] are on the level of the language considered. Here, patterns are not used, i.e. rewrite rules cannot impose structure on arguments of function symbols; moreover, no function symbol is allowed to appear in the right-hand side of rewrite rules.

5.1 Operations on types

We will now recall three operations on types from [2] that are needed in the definition of type assignment and are standard in intersection systems. Substitution is the operation that instantiates a type (i.e. that replaces type-variables by types). The operation of expansion replaces a type by the intersection of a number of copies of that type. The operation of lifting replaces a type by a larger one, in the sense of $\leq$.

These three operations are of use in Definition *5.7*, when we want to specify how, for a specific combinator, a type required by the context can be obtained from the type provided for that combinator by the environment (Definition *5.6*). It is possible to define type assignment with fewer or less powerful operations on types, but in order to obtain enough expressive power to prove Theorem *5.10 (i)*, all three operations are needed.

Definition 5.1 (TYPE-SUBSTITUTION) *i*) The *type-substitution* $(\varphi \mapsto \alpha) : \mathcal{T} \to \mathcal{T}$, that replaces occurrences of φ by α, where $\varphi \in \Phi$ and $\alpha \in \mathcal{T}_s \cup \{\omega\}$, is defined by:

$$\begin{array}{rcll}
(\varphi \mapsto \alpha)(\varphi) & = & \alpha & \\
(\varphi \mapsto \alpha)(\varphi') & = & \varphi' & \text{if } \varphi' \neq \varphi \\
(\varphi \mapsto \alpha)(\sigma \to \tau) & = & \omega & \text{if } (\varphi \mapsto \alpha)(\tau) = \omega \\
(\varphi \mapsto \alpha)(\sigma \to \tau) & = & (\varphi \mapsto \alpha)(\sigma) \to (\varphi \mapsto \alpha)(\tau) & \text{if } (\varphi \mapsto \alpha)(\tau) \neq \omega \\
(\varphi \mapsto \alpha)(\cap_{\underline{n}} \sigma_i) & = & (\varphi \mapsto \alpha)(\sigma'_1) \cap \cdots \cap (\varphi \mapsto \alpha)(\sigma'_m) & \text{where} \\
 & & \{\sigma'_1, \ldots, \sigma'_m\} = \{\sigma_i \in \{\sigma_i \,(i \in \underline{n})\} \mid (\varphi \mapsto \alpha)(\sigma_i) \neq \omega\} &
\end{array}$$

ii) The set of type-substitutions is closed under composition: if S_1 and S_2 are type-substitutions, then so is $S_1 \circ S_2$, where $S_1 \circ S_2(\sigma) = S_1(S_2(\sigma))$.

iii) $S(B) = \{x{:}S(\alpha) \mid x{:}\alpha \in B\}$.

iv) $S(\langle B, \sigma, E\rangle) = \langle S(B), S(\sigma), \{S(\rho) \mid \rho \in E\}\rangle$.

Note that the definition of substitution in an arrow type ensures that the resulting type is still in $\mathcal{T}$.

It is possible to define a notion of type-substitution that just replaces type variables by strict types (so where $\alpha \in \mathcal{T}_s$); using such a definition, we would be forced to use the extra operation of *covering* that deals with the introduction of ω (see also [3]; this operation is closely related to the covering relation of Definition *2.18*.) To keep the set of operations small, we have decided not to follow that direction here.

Our definition of expansion is inspired by the one given in [42] for the full intersection system in LC, we just need to make some minor changes to make sure that the type obtained is always in $\mathcal{T}$. For this, we have to check the last type-variable in arrow types (for a detailed discussion of the complexity of this operation, see [2]).

Definition 5.2 The *last type-variable* of a strict type, $last(\sigma)$, is defined by:

$$\begin{aligned} last(\varphi) &= \varphi \\ last(\sigma \to \tau) &= last(\tau) \end{aligned}$$

Definition 5.3 (EXPANSION) For every $\mu \in \mathcal{T}$ and $n \geq 2$, the pair $\langle \mu, n \rangle$ determines an *expansion* $Ex : \mathcal{T} \to \mathcal{T}$ which is computed with respect to $\langle B, \sigma, E \rangle$ as follows (where B is a basis, $\sigma \in \mathcal{T}$, and E is a finite set of types).

(*Affected variables*) : The set $\mathcal{V}_\mu(B, \sigma, E)$ of type-variables is defined by:

a) If φ occurs in μ, then $\varphi \in \mathcal{V}_\mu(B, \sigma, E)$.

b) If $last(\tau) \in \mathcal{V}_\mu(B, \sigma, E)$, with $\tau \in \mathcal{T}_s$ and τ (a subtype) in $\langle B, \sigma, E \rangle$, then for all type-variables φ that occur in τ: $\varphi \in \mathcal{V}_\mu(B, \sigma, E)$.

(*Renamings*) : Let $\mathcal{V}_\mu(B, \sigma, E) = \{\varphi_1, \ldots, \varphi_m\}$. Choose $m \times n$ different type-variables $\varphi_1^1, \ldots, \varphi_n^1, \ldots, \varphi_1^m, \ldots, \varphi_n^m$, such that each φ_i^j does not occur in $\langle B, \sigma, E \rangle$, for $i \in \underline{n}$ and $j \in \underline{m}$. Let S_i be such that $S_i(\varphi_j) = \varphi_i^j$.

(*Expansion of a type*) : $Ex(\tau)$ is defined by:

$$\begin{aligned} Ex(\tau_1 \cap \cdots \cap \tau_n) &= Ex(\tau_1) \cap \cdots \cap Ex(\tau_n) \\ Ex(\tau) &= S_1(\tau) \cap \cdots \cap S_n(\tau) && \text{if } last(\tau) \in \mathcal{V}_\mu(B, \sigma, E) \\ Ex(\varphi) &= \varphi && \text{if } \varphi \notin \mathcal{V}_\mu(B, \sigma, E) \\ Ex(\sigma \to \rho) &= Ex(\sigma) \to Ex(\rho) && \text{if } last(\rho) \notin \mathcal{V}_\mu(B, \sigma, E) \end{aligned}$$

(*Expansion of* B) : $Ex(B) = \{x{:}Ex(\rho) \mid x{:}\rho \in B\}$.

(*Expansion of* $\langle B, \sigma, E \rangle$) : $Ex(\langle B, \sigma, E \rangle) = \langle Ex(B), Ex(\sigma), \{Ex(\rho) \mid \rho \in E\} \rangle$.

When an expansion operation Ex is applied to a type τ without specifying $\langle B, \sigma, E \rangle$ we assume that the expansion is computed with respect to $\langle \emptyset, \tau, \emptyset \rangle$.

Definition 5.4 (LIFTING) A *lifting* L is an operation denoted by $\langle \langle B_0, \tau_0 \rangle, \langle B_1, \tau_1 \rangle \rangle$, a pair of pairs such that $\tau_0 \leq \tau_1$ and $B_1 \leq B_0$, and is defined by:

$$\begin{aligned} L(\sigma) &= \tau_1 \text{ if } \sigma = \tau_0 & \qquad L(B) &= B_1 \text{ if } B = B_0 \\ L(\sigma) &= \sigma \text{ otherwise} & \qquad L(B) &= B \text{ otherwise} \end{aligned}$$

Definition 5.5 (CHAINS OF OPERATIONS ON TYPES) A *chain on types* is an object $[O_1, \ldots, O_n]$, where each O_i is an operation of type-substitution, expansion or lifting, and

$$[O_1, \ldots, O_n](\sigma) = O_n(\cdots(O_1(\sigma))\cdots)$$

We will use $*$ to denote the operation of concatenation of chains.

5.2 Type assignment

To complete the definition of type assignment, we present now the type assignment rules that are used to assign types in $\mathcal{T}$ to terms and combinator rules. In order to type the combinators, we use an environment that provides a type in $\mathcal{T}_s$ for every $C \in \mathcal{C}$, and use chains of operations to obtain the type for an occurrence of the combinator from the type provided for it by the environment.

Definition 5.6 (ENVIRONMENT) Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a CS.

i) An *environment for* $\mathbf{C}$ is a mapping $\mathcal{E} : \mathcal{C} \to \mathcal{T}_s$.

ii) For $C \in \mathcal{C}$, $\tau \in \mathcal{T}_s$, and $\mathcal{E}$ an environment, the environment $\mathcal{E}[C \mapsto \tau]$ is defined by:

$$\begin{aligned} \mathcal{E}[C \mapsto \tau]\,(D) &= \tau \qquad \text{if } D = C \\ \mathcal{E}[C \mapsto \tau]\,(D) &= \mathcal{E}(D) \ \text{ otherwise} \end{aligned}$$

Since an environment $\mathcal{E}$ maps all $C \in \mathcal{C}$ to types in $\mathcal{T}_s$, no combinator is mapped to ω.

We define now type assignment on terms and combinator rules.

Definition 5.7 (TYPE ASSIGNMENT) Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a CS and $\mathcal{E}$ an environment for $\mathbf{C}$.

i) *Type assignment* for terms in $\mathcal{T}(\mathcal{C}, \mathcal{V})$ and *derivations* are defined by the following natural deduction system (where all types displayed are in $\mathcal{T}_s$, except for τ in rules $(\leq)$ and $(\to E)$):

$$(\mathcal{E}) : \frac{}{B \vdash_{\mathcal{E}} C{:}\sigma}\ (\exists Ch\,[Ch(\mathcal{E}(C)) = \sigma]) \qquad (\to E) : \frac{B \vdash_{\mathcal{E}} t_1{:}\tau \to \sigma \quad B \vdash_{\mathcal{E}} t_2{:}\tau}{B \vdash_{\mathcal{E}} t_1 t_2{:}\sigma}$$

$$(\cap I) : \frac{B \vdash_{\mathcal{E}} t{:}\sigma_1 \quad \cdots \quad B \vdash_{\mathcal{E}} t{:}\sigma_n}{B \vdash_{\mathcal{E}} t{:}\cap_{\underline{n}}\sigma_i}\ (n \geq 0) \qquad (\leq) : \frac{}{B \vdash_{\mathcal{E}} x{:}\sigma}\ (x{:}\tau \in B, \tau \leq \sigma)$$

If $B \vdash_{\mathcal{E}} t{:}\sigma$ is derivable using a derivation D, we write $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$. We write $B \vdash_{\mathcal{E}} t{:}\sigma$ to express that there exists a derivation D such that $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$, and $\vdash_{\mathcal{E}} t{:}\sigma$ when $\emptyset \vdash_{\mathcal{E}} t{:}\sigma$. We will write $B \vdash^{\omega}_{\mathcal{E}} t{:}\sigma$ if ω is not used in the derivation.

ii) Let $C \in \mathcal{C}$, with *arity*$(C) = n$. The combinator rule $C\,x_1 \cdots x_n \to r \in \mathbf{R}$ *is typeable with respect to* $\mathcal{E}$, if there are $\sigma_i\,(i \in \underline{n}) \in \mathcal{T}$ and $\sigma \in \mathcal{T}_s$, such that $\mathcal{E}(C) = \sigma_1 \to \cdots \to \sigma_n \to \sigma$, and $\{\overrightarrow{x_i{:}\sigma_i}\} \vdash_{\mathcal{E}} r{:}\sigma$.

iii) $\mathbf{C}$ *is typeable with respect to* $\mathcal{E}$, if every rule in $\mathbf{R}$ is typeable with respect to $\mathcal{E}$.

At first sight, the formulation '*is typeable with respect to* $\mathcal{E}$' might seem a restriction on the class of systems that are considered in this section, but it is not. Notice that an environment just maps combinators to types, without regard for the structure of their rewrite rules. The condition is added above just to ascertain that the type provided by the environment actually makes sense, and respects the structure of the rules involved.

The reason not to allow environments to provide types outside of $\mathcal{T}_s$ is purely practical, to obtain easier definitions. Notice that it is possible to derive an intersection type for a combinator, using rule $(\mathcal{E})$ a number of times, followed by $(\cap I)$.

Example 5.8 The rules of CL (see Example *4.4*) are typeable with respect to the environment $\mathcal{E}_{\mathrm{CL}}$:

$$\begin{aligned} \mathcal{E}_{\mathrm{CL}}(\mathsf{S}) &= (\varphi_1 \to \varphi_2 \to \varphi_3) \to (\varphi_4 \to \varphi_2) \to \varphi_1 \cap \varphi_4 \to \varphi_3 \\ \mathcal{E}_{\mathrm{CL}}(\mathsf{K}) &= \varphi_5 \to \omega \to \varphi_5 \\ \mathcal{E}_{\mathrm{CL}}(\mathsf{I}) &= \varphi_6 \to \varphi_6 \end{aligned}$$

The term SKSI can be typed with the type $\alpha \to \alpha$ with respect to $\mathcal{E}_{\mathrm{CL}}$: take

$$\begin{aligned} Ch_1 &= [(\varphi_1 \mapsto \alpha \to \alpha), (\varphi_2 \mapsto \omega), (\varphi_3 \mapsto \alpha \to \alpha), (\varphi_4 \mapsto \omega)] \\ Ch_2 &= [(\varphi_5 \mapsto \alpha \to \alpha)] \\ Ch_3 &= [(\varphi_6 \mapsto \alpha)] \end{aligned}$$

then (notice that $Ch_1\,(\varphi_4{\to}\varphi_2) = \omega$ and $Ch_1\,(\varphi_1{\cap}\varphi_4) = \alpha{\to}\alpha$)

$$\begin{aligned} Ch_1\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{S})) &= ((\alpha{\to}\alpha){\to}\omega{\to}\alpha{\to}\alpha){\to}\omega{\to}(\alpha{\to}\alpha){\to}\alpha{\to}\alpha \\ Ch_2\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{K})) &= (\alpha{\to}\alpha){\to}\omega{\to}\alpha{\to}\alpha \\ Ch_3\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{I})) &= \alpha{\to}\alpha \end{aligned}$$

and

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{S}:Ch_1\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{S}))\quad \vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{K}:Ch_2\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{K}))}{\vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{SK}:\omega{\to}(\alpha{\to}\alpha){\to}\alpha{\to}\alpha}\quad \vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{S}:\omega}{\vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{SKS}:(\alpha{\to}\alpha){\to}\alpha{\to}\alpha}\quad \vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{I}:Ch_3\,(\mathcal{E}_{\mathrm{CL}}(\mathsf{I}))}{\vdash_{\mathcal{E}_{\mathrm{CL}}}\mathsf{SKSI}:\alpha{\to}\alpha}$$

The definition of type assignment on CS as presented in this section allows for the formulation of a precise relation between types assignable to terms, and those assignable to equivalent lambda terms. In fact, a result similar to part of the following property has already been proved in [21].

The relation between type assignment in LC and that in **C** (restricted to CL with the environment $\mathcal{E}_{\mathrm{CL}}$) is very strong, as the following theorem shows. To understand that this property is not straightforward is shown by the following example: take $\vdash \lambda x.x:\alpha{\to}\alpha$ and notice that $[\![\lambda x.x]\!]_{\mathrm{CL}} = \mathsf{I}$. If $\mathcal{E}\,(\mathsf{I}) = (\alpha{\to}\alpha){\to}\alpha{\to}\alpha$, then it is not possible to assign $\alpha{\to}\alpha$ to I in $\vdash_{\mathcal{E}}$ (see also Section *7*).

However, we can show the following two results for CS equipped with principal environments, as defined below.

Definition 5.9 Let $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS. The environment $\mathcal{E}$ is called *principal for* **C**, if for all $C \in \mathcal{C}$, $\mathcal{E}\,(C)$ is the principal type for $\langle C\rangle_\lambda^{\mathbf{C}}$ in $\vdash$.[1]

Theorem 5.10 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS.

i) If $\mathcal{E}$ *is principal for* **C**, *then* $B \vdash \langle t\rangle_\lambda^{\mathbf{C}}{:}\sigma$ *implies* $B \vdash_{\mathcal{E}} t{:}\sigma$.

ii) $B \vdash_{\mathcal{E}} t{:}\sigma$ *implies* $B \vdash \langle t\rangle_\lambda^{\mathbf{C}}{:}\sigma$.

Proof: Assume (without loss of generality), that $\sigma \in \mathcal{T}_s$.

i) By induction on the structure of terms in $\mathcal{T}(\mathcal{C},\mathcal{V})$. The only case that needs attention is that of $t = C \in \mathcal{C}$, so $B \vdash \langle C\rangle_\lambda^{\mathbf{C}}{:}\sigma$. Since $\mathcal{E}$ is principal for **C**, $\mathcal{E}(C)$ is the principal type for $\langle C\rangle_\lambda^{\mathbf{C}}$ in $\vdash$ and there exists (see [2]) a chain of operations *Ch* such that $Ch(\mathcal{E}(C)) = \sigma$. But then $B \vdash_{\mathcal{E}} C{:}\sigma$ by rule $(\mathcal{E})$.

ii) By induction on the definition of $\langle\ \rangle_\lambda^{\mathbf{C}}$; the only alternative that needs consideration is that of $t = C \in \mathcal{C}$, and then the last rule in the derivation for $B \vdash_{\mathcal{E}} t{:}\sigma$ is $(\mathcal{E})$. Then there is a chain *Ch* such that $Ch(\mathcal{E}(C)) = \sigma$. Let $Cx_1\cdots x_n \to r$ be the rule for C. Then, by Definition *5.7(ii)*, there are $\tau_j\ (j \in \underline{n}) \in \mathcal{T}$ and $\tau \in \mathcal{T}_s$, such that $\{x_1{:}\tau_1,\ldots,x_n{:}\tau_n\} \vdash_{\mathcal{E}} r{:}\tau$ and $\mathcal{E}(C) = \tau_1{\to}\cdots{\to}\tau_n{\to}\tau$. Then, by induction, $\{x_1{:}\tau_1,\ldots,x_n{:}\tau_n\} \vdash r{:}\tau$ (notice that $\langle r\rangle_\lambda^{\mathbf{C}} = r$). Then, by rule $(\to\!I)$ of $\vdash$,

[1] Since for every $l \to r \in \mathbf{R}$, r is in normal form, not containing combinators, it is possible to define the notion of principal environment directly for CS, without side-stepping to LC, but that would significantly increase the complexity of the proofs of this section. It would not affect any of the results; in fact, the definition above would become a provable property.

$\vdash \lambda x_1 \ldots x_n.r\!:\!\tau_1 \!\rightarrow\! \cdots \!\rightarrow\! \tau_n \!\rightarrow\! \tau$; since $\vdash$ is closed for all three operations of substitution, expansion, and lifting (see [3]), we also have $\vdash \lambda x_1 \ldots x_n.r\!:\!\sigma$', so $\vdash \langle C \rangle_\lambda^{\mathbf{C}}\!:\!\sigma$. ■

5.3 Subject reduction

In this section we will show that the notion of type assignment defined here on CS satisfies the subject reduction property (Theorem *5.17*). In order to achieve this, we need that the three operations (type-substitution, expansion, and lifting) defined in the previous section can be applied to type-derivations, and are sound (the result is a well-defined derivation); the details of this are skipped here (for a complete version, see [9]). We will also show that the type assignment rule ($\mathcal{E}$) is sound in the following sense: if there is an operation *O* such that $O(\mathcal{E}(C)) = \sigma$, then, for every type $\tau \in \mathcal{T}_s$ such that $\sigma \leq \tau$, the combinator rule for C is typeable with respect to the changed environment $\mathcal{E}[C \mapsto \tau]$.

Proposition 5.11 (SOUNDNESS OF TYPE-SUBSTITUTION) *Let S be a type-substitution.*

i) If $B \vdash_{\mathcal{E}} t\!:\!\sigma$, then $S(B) \vdash_{\mathcal{E}} t\!:\!S(\sigma)$.

ii) If $Cx_1 \cdots x_n \rightarrow r$ is a rule typeable with respect to the environment $\mathcal{E}$, and $S(\mathcal{E}(C)) \neq \omega$, then it is typeable with respect to $\mathcal{E}[C \mapsto S(\mathcal{E}(C))]$.

The following essentially shows that lifting is sound:

Lemma 5.12 i) If $B \vdash_{\mathcal{E}} t\!:\!\sigma$ and $B' \leq B$, then $B' \vdash_{\mathcal{E}} t\!:\!\sigma$.

ii) If $B \vdash_{\mathcal{E}} t\!:\!\sigma$ and $\sigma \leq \tau$, then $B \vdash_{\mathcal{E}} t\!:\!\tau$.

iii) If $B \vdash_{\mathcal{E}}^{\omega} t\!:\!\sigma$, $\sigma \leq \tau$, and τ is ω-free, then $B \vdash_{\mathcal{E}}^{\omega} t\!:\!\tau$.

Proposition 5.13 (SOUNDNESS OF LIFTING) *Let L be a lifting.*

i) If $B \vdash_{\mathcal{E}} t\!:\!\rho$, then $L(B) \vdash_{\mathcal{E}} t\!:\!L(\rho)$.

ii) If $Cx_1 \cdots x_n \rightarrow r$ is a combinator rule, typeable with respect to $\mathcal{E}$, and $L(\mathcal{E}(C)) \in \mathcal{T}_s$, then it is typeable with respect to $\mathcal{E}[C \mapsto L(\mathcal{E}(C))]$.

Proposition 5.14 (SOUNDNESS OF EXPANSION) *Let Ex be an expansion operation determined by $\langle \mu, n \rangle$, such that $Ex\langle B, \sigma, E \rangle = \langle B', \sigma', E' \rangle$.*

i) If $B \vdash_{\mathcal{E}} t\!:\!\sigma$ using a set E of types for the occurrences of combinators in t, then $B' \vdash_{\mathcal{E}} t\!:\!\sigma'$.

ii) If $Cx_1 \cdots x_n \rightarrow r$ is a rule, typeable with respect to $\mathcal{E}$, and $Ex(\mathcal{E}(C)) = \cap_{\underline{m}} \tau_j \in \mathcal{T}$ $(m \geq 1)$, then, for every $j \in \underline{m}$, the rule is typeable with respect to $\mathcal{E}[C \mapsto \tau_j]$.

We then have:

Theorem 5.15 (SOUNDNESS OF CHAINS) *i) The set of derivations is closed under chains of operations.*

ii) Let $l \rightarrow_C r$ be a combinator rule typeable with respect to the environment $\mathcal{E}$. If $Ch(\mathcal{E}(C))$ $= \tau \in \mathcal{T}$, then, for every $\mu \in \mathcal{T}_s$ such that $\tau \leq \mu$, C is typeable with respect to $\mathcal{E}[C \mapsto \mu]$.

Proof: By Propositions *5.11*, *5.13*, and *5.14*. ■

Using this soundness result, we will now show that the notion of type assignment as defined in this section satisfies the subject reduction property: if $B \vdash_{\mathcal{E}} t\!:\!\sigma$, and t can be rewritten to t', then $B \vdash_{\mathcal{E}} t'\!:\!\sigma$. Of course, this result can be obtained through the mappings $[\![\ ]\!]_{\mathbf{C}}$ and

$\langle\ \rangle_\lambda^{\mathbf{C}}$, using the relations between the systems mentioned in the previous section, but only for combinatory complete CS and principal environments. For other CS, we must give a direct proof, for which we need the following result.

Lemma 5.16 (TERM-SUBSTITUTION LEMMA) *i) If $B \vdash_{\mathcal{E}} t{:}\sigma$, then, for every term-substitution* R *and basis B', if for every $x{:}\tau \in B$, $B' \vdash_{\mathcal{E}} x^{\mathrm{R}}{:}\tau$, then $B' \vdash_{\mathcal{E}} t^{\mathrm{R}}{:}\sigma$.*

ii) Let $Cx_1 \cdots x_n \to r$ be a combinator rule, typeable with respect to $\mathcal{E}$. For every term-substitution R*, basis B and type μ: if $B \vdash_{\mathcal{E}} (Cx_1 \cdots x_n)^{\mathrm{R}}{:}\mu$, then $B \vdash_{\mathcal{E}} r^{\mathrm{R}}{:}\mu$.*

Proof: *i*) By induction on $\vdash_{\mathcal{E}}$.

$(\leq)$: Then $t = x$. Then there is $x{:}\tau \in B$, such that $\tau \leq \sigma$. Then, by Theorem *5.13*, $B' \vdash_{\mathcal{E}} x^{\mathrm{R}}{:}\tau$ implies $B' \vdash_{\mathcal{E}} x^{\mathrm{R}}{:}\sigma$.

$(\mathcal{E})$: Then $t = C$. Immediate, since $C^{\mathrm{R}} = C$, and $C{:}\sigma$ does not depend on the basis.

$(\to E),\ (\cap I)$: By induction.

ii) If $Cx_1 \cdots x_n \to r$ is a typeable combinator rule, then by Definition *5.7(ii)*, there are $\sigma_i\,(i \in \underline{n}), \sigma$, such that $\mathcal{E}(C) = \sigma_1 \to \cdots \to \sigma_n \to \sigma$ and $\{x_i{:}\sigma_i\} \vdash_{\mathcal{E}} r{:}\sigma$. Also, $(Cx_1 \cdots x_n)^{\mathrm{R}} = Cx_1{}^{\mathrm{R}} \cdots x_n{}^{\mathrm{R}}$. Since $B \vdash_{\mathcal{E}} Cx_1{}^{\mathrm{R}} \cdots x_n{}^{\mathrm{R}}{:}\mu$, there are two cases:

$(\mu \in \mathcal{T}_s)$: then there are $\mu_i\,(i \in \underline{n})$, and a chain *Ch* such that $Ch(\mathcal{E}(C)) = \mu_1 \to \cdots \to \mu_n \to \mu$, and, for $i \in \underline{n}$, $B \vdash_{\mathcal{E}} x_i{}^{\mathrm{R}}{:}\mu_i$. Since $\{x_i{:}\sigma_i\} \vdash_{\mathcal{E}} r{:}\sigma$, we have, by Theorem *5.15(i)*, $\{x_i{:}\mu_i\} \vdash_{\mathcal{E}} r{:}\mu$. Then, by part *(i)*, also $B \vdash_{\mathcal{E}} r^{\mathrm{R}}{:}\mu$.

$(\mu = \rho_1 \cap \ldots \cap \rho_n)$: we apply the above reasoning to each ρ_i and apply $(\cap I)$. ■

Using this result, the following becomes easy.

Theorem 5.17 (SUBJECT REDUCTION) *Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a* CS*. For all $t, t' \in \mathcal{T}(\mathcal{C}, \mathcal{V})$: if $B \vdash_{\mathcal{E}} t{:}\sigma$ and $t \to^* t'$, then $B \vdash_{\mathcal{E}} t'{:}\sigma$.*

Proof: By induction on the length of the reduction path; the case of length 1 is proved by induction on the structure of t. Of this double induction, only the case that t itself is the term-substitution instance of a left-hand side of a combinator rule is of interest; all other cases are straightforward. Then, let $C \in \mathcal{C}$ and term-substitution R be such that $l \to_C r$, $t = l^{\mathrm{R}}$, and $t' = r^{\mathrm{R}}$. The result follows from Lemma *5.16(ii)*. ■

One should remark that a subject expansion theorem, i.e. the converse of the subject reduction result:

$$\textit{If } B \vdash_{\mathcal{E}} t{:}\sigma, \textit{ and } t' \to t, \textit{ then } B \vdash_{\mathcal{E}} t'{:}\sigma,$$

does not hold in general. Take for example the following CS, that is typeable with respect to the given environment

$$\begin{array}{rcl@{\qquad}rcl} \mathsf{K}xy & \to & x & \mathcal{E}(\mathsf{K}) & = & \varphi_1 \to \omega \to \varphi_1 \\ \mathsf{I}x & \to & x & \mathcal{E}(\mathsf{I}) & = & (\varphi_2 \to \varphi_2) \to \varphi_2 \to \varphi_2 \end{array}$$

The term IK reduces to the (head-)normal form K, but can only be typed by ω with respect to $\mathcal{E}$. Of course, $(\varphi_2 \to \varphi_2) \to \varphi_2 \to \varphi_2$ is not the principal type for $\langle \mathsf{I} \rangle_\lambda^{\mathrm{CL}}$ in $\vdash$. In fact, we have the following result:

Theorem 5.18 (SUBJECT EXPANSION) *Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a* CS*, and $\mathcal{E}$ be principal for* **C***, then, for all $t, t' \in \mathcal{T}(\mathcal{C}, \mathcal{V})$: if $B \vdash_{\mathcal{E}} t{:}\sigma$ and $t' \to t$, then $B \vdash_{\mathcal{E}} t'{:}\sigma$.*

Proof: If $B \vdash_{\mathcal{E}} t{:}\sigma$, then by Lemma *5.10(ii)*, also $B \vdash \langle t \rangle_\lambda^{\mathbf{C}} : \sigma$. Since $t' \to t$, by Proposition

4.8 also $\langle t' \rangle_\lambda^{\mathbf{C}} \twoheadrightarrow_\beta \langle t \rangle_\lambda^{\mathbf{C}}$. Since $\vdash$ is closed for β-expansion, we have $B \vdash \langle t' \rangle_\lambda^{\mathbf{C}} : \sigma$. Then, by Theorem *5.10(i)*, we have $B \vdash_{\mathcal{E}} t' : \sigma$. ■

5.4 Derivation reduction is strongly normalising

We will skip the precise definition of derivation reduction here, and just highlight (in simplified form) the main steps.

The notion of reduction of derivations depends, of course, on a notion of *derivation substitution*, of which the principle is sketched by the following.

Assume $\mathrm{D}_1 :: B, x{:}\tau \vdash t : \sigma$ and $\mathrm{D}_2 :: B \vdash u : \tau$. The result of substituting D_2 in D_1 for $x{:}\tau$ is defined in a way similar to the one discussed above for LC. Notice that there are (strict) types $\rho_j \, (j \in \underline{m})$ such that, for every ρ_j, in D_1, there exists a sub-derivation $\mathrm{D}_x^j :: B, x{:}\tau \vdash x : \rho_j$. Then, for all $j \in \underline{m}$, $\tau \leq \rho_j$ and, by Proposition *5.12*, there exists a derivation $\mathrm{D}_j :: B \vdash N : \rho_j$. Then a derivation for

$$\mathrm{D}_1[\mathrm{D}_2/x{:}\tau] :: B \vdash t[u/x] : \sigma$$

can be obtained by replacing, in D_1, for every $j \in \underline{m}$, the sub-derivation D_x^j by D_j.

So, assume now that $t = (C x_1 \cdots x_n)^{\mathrm{R}} = C t_1 \cdots t_n$ and that there is a combinator rule $C x_1 \cdots x_n \to r$. Then D has the form:

$$\dfrac{\dfrac{\dfrac{}{B \vdash_{\mathcal{E}} C{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma} \quad \dfrac{\mathrm{D}_1}{B \vdash_{\mathcal{E}} t_1{:}\sigma_1}}{B \vdash_{\mathcal{E}} C t_1{:}\sigma_2 \to \cdots \to \sigma_n \to \sigma} \quad \dfrac{\mathrm{D}_n}{B \vdash_{\mathcal{E}} t_n{:}\sigma_n}}{\vdots}}{B \vdash_{\mathcal{E}} C t_1 \cdots t_n{:}\sigma}$$

Then, by Definition *5.7(ii)*, there exists $\mathrm{D}_0 :: \{\overrightarrow{x_i{:}\sigma_i}\} \vdash_{\mathcal{E}} r{:}\sigma$. Let $\mathrm{R} = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ and $t' = r^{\mathrm{R}}$, then D reduces to

$$\mathrm{D}' = \mathrm{D}_0 \, [\mathrm{D}_1/x_1{:}\sigma_1, \ldots, \mathrm{D}_n/x_n{:}\sigma_n] :: B \vdash_{\mathcal{E}} t' : \sigma,$$

which is a well-defined derivation.

Remark that each derivation redex corresponds to a term redex (because of the presence of ω, the converse does not hold), and that if $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$ reduces to $\mathrm{D}' :: B \vdash_{\mathcal{E}} t'{:}\sigma$, then t reduces to t'.

As shown in [9] (not repeated here), derivations in the restricted type assignment system are strongly normalizable with respect to the notion of reduction suggested here.

Theorem 5.19 (STRONG NORMALISATION) *If* $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$, *then SN*$(\mathrm{D})$. ■

6 Approximants

Now we will develop, essentially following [46] (see also [10]), a notion of approximant for combinator terms. As in Section *2.1* this will be done by introducing $\bot$ into the definition of terms.

Definition 6.1 (COMBINATOR TERMS WITH $\bot$) Let $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS.

i) The set $T(\mathcal{C},\mathcal{X},\bot)$ is defined by the following grammar:

$$t ::= \bot \mid x \mid C \mid \mathit{Ap}\,(t_1,t_2)$$

ii) The notion of rewriting of Definition *4.3* extends naturally to terms in $T(\mathcal{C},\mathcal{X},\bot)$, and we will use the same symbol ' $\rightarrow_{\mathbf{R}}$ ' to denote the rewriting relation induced by **C** on $T(\mathcal{C},\mathcal{X},\bot)$.

The relation $\sqsubseteq$ on terms, as given in the following definition, takes $\bot$ to be the smallest term.

Definition 6.2 *i*) We define the relation $\sqsubseteq$ on $T(\mathcal{C},\mathcal{X},\bot)$ inductively by:

$$\begin{aligned} \bot &\sqsubseteq t, \\ t &\sqsubseteq t, \\ t_1 \sqsubseteq u_1 \;\&\; t_2 \sqsubseteq u_2 &\iff t_1 t_2 \sqsubseteq u_1 u_2. \end{aligned}$$

ii) t and u are called *compatible* if there exists a v such that $t \sqsubseteq v$ and $u \sqsubseteq v$.

We will now come to the definition of approximate normal forms and of direct approximants. The general idea is that a direct approximant of a term t is constructed out of t by replacing all redexes and potential redexes in t by $\bot$ (a potential redex is a sub-term that *could* be a redex if $\bot$ were to be replaced by an appropriate term).

Definition 6.3 (APPROXIMATE NORMAL FORMS) Let $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS.

i) $\mathcal{A}_{\mathcal{C}}$, the set of *approximate normal forms* of $T(\mathcal{C},\mathcal{X},\bot)$, ranged over by a, is defined by:

$$a ::= \bot \mid x a_1 \cdots a_n \;(n \geq 0) \mid C a_1 \cdots a_n \;(n < \mathit{arity}(C)).$$

ii) $\mathcal{DA}_{\mathcal{C}}(t)$, the *direct approximant of* t with respect to **C** is defined by:

$$\begin{aligned} \mathcal{DA}_{\mathcal{C}}(x) &= x \\ \mathcal{DA}_{\mathcal{C}}(C) &= C \\ \mathcal{DA}_{\mathcal{C}}(t_1 t_2) &= \bot \text{ if } \mathcal{DA}_{\mathcal{C}}(t_1) = \bot \quad \text{or} \\ &\qquad \mathcal{DA}_{\mathcal{C}}(t_1) = C a_1 \cdots a_n \text{ and } \mathit{arity}(C) = n{+}1 \\ &= \mathcal{DA}_{\mathcal{C}}(t_1)\,\mathcal{DA}_{\mathcal{C}}(t_2) \text{otherwise} \end{aligned}$$

Notice that every normal form in $\mathcal{T}(\mathcal{C},\mathcal{V})$ is also an approximate normal form.

For $\sqsubseteq$, the following properties hold:

Lemma 6.4 *i)* $t \sqsubseteq u \sqsubseteq v \Rightarrow t \sqsubseteq v$.
ii) t *is a head-normal form* $\iff \exists a \in \mathcal{A}_{\mathcal{C}}\,[a \sqsubseteq t \;\&\; a \neq \bot]$.
iii) If $a \in \mathcal{A}_{\mathcal{C}}$ *and* $a \sqsubseteq t$, *then* $a \sqsubseteq \mathcal{DA}_{\mathcal{C}}(t)$.

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

The relation between reduction and $\sqsubseteq$ is expressed by:

Lemma 6.5 *i)* $a \in \mathcal{A}_{\mathcal{C}} \;\&\; v \rightarrow^* w \;\&\; a \sqsubseteq v \Rightarrow a \sqsubseteq w$.
ii) $t_0 \sqsubseteq t \;\&\; t_0 \rightarrow t_1 \Rightarrow \exists t'[t \rightarrow t' \;\&\; t_1 \sqsubseteq t']$.

Proof: By induction on the structure of terms. ■

We will now introduce, similar to Definition *2.5*, a notion of 'join' on terms containing $\bot$, that is of use in the proof of Lemma *6.13*.

Definition 6.6 On $T(\mathcal{C},\mathcal{X},\bot)$, the partial mapping $\sqcup : T(\mathcal{C},\mathcal{X},\bot) \times T(\mathcal{C},\mathcal{X},\bot) \to T(\mathcal{C},\mathcal{X},\bot)$ is defined by:

$$\begin{aligned} \bot \sqcup t = t \sqcup \bot &= t \\ t \sqcup t &= t \\ (t_1\, t_2) \sqcup (u_1\, u_2) &= (t_1 \sqcup u_1)(t_2 \sqcup u_2) \end{aligned}$$

Again, we will use our more general definition only on terms that are compatible.

The following lemma shows that $\sqcup$ acts as least upper bound for compatible terms.

Lemma 6.7 If $t_1 \sqsubseteq t$ and $t_2 \sqsubseteq t$, then $t_1 \sqcup t_2$ is defined, and: $t_1 \sqsubseteq t_1 \sqcup t_2$, $t_2 \sqsubseteq t_1 \sqcup t_2$, and $t_1 \sqcup t_2 \sqsubseteq t$.

Proof: By induction on the structure of terms. ■

Approximants of terms are defined by:

Definition 6.8 (APPROXIMANTS) $\mathcal{A}_\mathcal{C}(t) = \{a \in \mathcal{A}_\mathcal{C} \mid \exists u\, [t \to^* u \;\&\; a \sqsubseteq u]\}$ is the *set of approximants of t*.

Notice that we could have used '$a \sqsubseteq \mathcal{DA}_\mathcal{C}(u)$' as well.

In Section *7*, using this definition, we will define a semantics for CS, and we will need the following properties relating approximants and reduction.

Lemma 6.9 i) $t \to^* t' \Rightarrow \mathcal{A}_\mathcal{C}(t) = \mathcal{A}_\mathcal{C}(t')$.
ii) $a, a' \in \mathcal{A}_\mathcal{C}(t) \Rightarrow a \sqcup a' \in \mathcal{A}_\mathcal{C}(t)$.

Proof: *i)* $(\subseteq)$:

$$\begin{array}{lll} t \to^* t' \;\&\; a \in \mathcal{A}_\mathcal{C}(t) & \Rightarrow & \\ t \to^* t' \;\&\; \exists v\, [t \to^* v \;\&\; a \sqsubseteq v] & \Rightarrow & (4.5) \\ \exists v, w\, [t \to^* v \;\&\; v \to^* w \;\&\; t' \to^* w \;\&\; a \sqsubseteq v] & \Rightarrow & (6.5(i)) \\ \exists w\, [t' \to^* w \;\&\; a \sqsubseteq w] & \Rightarrow & a \in \mathcal{A}_\mathcal{C}(t') \end{array}$$

$(\supseteq)$:

$$\begin{array}{lll} t \to^* t' \;\&\; a \in \mathcal{A}_\mathcal{C}(t') & \Rightarrow & \\ t \to^* t' \;\&\; \exists v\, [t' \to^* v \;\&\; a \sqsubseteq v] & \Rightarrow & \\ \exists v\, [t \to^* v \;\&\; a \sqsubseteq v] & \Rightarrow & a \in \mathcal{A}_\mathcal{C}(t) \end{array}$$

ii)

$$\begin{array}{ll} a \in \mathcal{A}_\mathcal{C}(t) \;\&\; a' \in \mathcal{A}_\mathcal{C}(t) & \Rightarrow (6.8) \\ \exists u, u'\, [t \to^* u \;\&\; a \sqsubseteq u \;\&\; t \to^* u' \;\&\; a' \sqsubseteq u'] & \Rightarrow (4.5 \;\&\; 6.5(i)) \\ \exists u, u', v\, [t \to^* u \to^* v \;\&\; t \to^* u' \to^* v \;\&\; a \sqsubseteq v \;\&\; a' \sqsubseteq v] & \Rightarrow (6.7) \\ \exists v\, [t \to^* v \;\&\; a \sqcup a' \sqsubseteq v] & \Rightarrow a \sqcup a' \in \mathcal{A}_\mathcal{C}(t) \end{array}$$

Lemma 6.10 If $\mathcal{A}_\mathcal{C}(t) = \{\bot\}$, then t is unsolvable.

Proof: If $\mathcal{A}_\mathcal{C}(t) = \{\bot\}$, then, for all v such that $t \to^* v$, and $a \in \mathcal{A}_\mathcal{C}$, if $a \sqsubseteq v$, then $a = \bot$. So, in particular, there is no v such that $t \to^* v$ and v is of the shape $x\, a_1 \cdots a_n$, with $(n \geq 0)$ or $C\, a_1 \cdots a_n$ with $(n < \mathit{arity}(C))$, since otherwise $x \bot \cdots \bot \sqsubseteq v$ or $C \bot \cdots \bot \sqsubseteq v$. Therefore, t does not reduce to a term in head normal form: it is unsolvable. ■

The following result is crucial for the proof of Lemma *7.3*:

Lemma 6.11 *Let* $t_1, t_2 \in \mathcal{T}(\mathcal{C}, \mathcal{V})$, $a \in \mathcal{A}_{\mathcal{C}}(t_1 t_2)$, *then there exist* $a_1 \in \mathcal{A}_{\mathcal{C}}(t_1)$, $a_2 \in \mathcal{A}_{\mathcal{C}}(t_2)$ *and* u' *such that* $a_1 a_2 \to^* u'$ *and* $a \sqsubseteq u'$.

Proof: The case $a = \bot$ is trivial. For $a \neq \bot$: assume $t_1 t_2 \to^* u$ and $a \sqsubseteq u$, then either:

i) $u = u_1 u_2$, and $t_j \to^* u_j$, for $j = 1, 2$. Since $a \sqsubseteq u_1 u_2$ and $a \neq \bot$, there are a_1, a_2 such that $a = a_1 a_2$, and $a_j \sqsubseteq u_j$, for $j = 1, 2$. Notice that $a_1 a_2 \in \mathcal{A}_{\mathcal{C}}$, and take $u' = a$.

ii) There exist $C, p_1, \ldots, p_n$ such that $C x_1 \cdots x_n \to r$,

$$t_1 t_2 \to^* C p_1 \cdots p_n \to r^{\vec{p}} \to^* u,$$

and none of the reductions in the first part of this sequence take place at the root position. Since some of the reductions that take place after contracting the redex $C p_1 \cdots p_n$ are in fact residuals of redexes already occurring in $p_1, \ldots, p_n$, we can take the reduction sequence that first contracts all redexes (and their residuals) that already occur in $p_1, \ldots, p_n$. Then, since the rewrite system is orthogonal (i.e. rules are left linear and without superpositions), there exists $p'_1, \ldots, p'_n$ and v such that

$$t_1 t_2 \to^* C p_1 \cdots p_n \to^* C p'_1 \cdots p'_n \to r^{\vec{p'}} \to^* v \text{ and } u \to^* v$$

and in the reduction sequence $r^{\vec{p'}} \to^* v$ we mimic $r^{\vec{p}} \to^* u$, but only contract redexes that are created *after* the redex $C p'_1 \cdots p'_n$ was contracted. Take $a_i = \mathcal{DA}_{\mathcal{C}}(p'_i)$, for $i \in \underline{n}$, then the redexes that are erased have no relevance to the sequence $r^{\vec{p'}} \to^* v$; moreover, there is only one redex in $C a_1 \cdots a_n$, being that term itself, and both $C a_1 \cdots a_{n-1}$ and a_n are in $\mathcal{A}_{\mathcal{C}}$. Notice that $t_1 \to^* C p'_1 \cdots p'_{n-1}$, and $C a_1 \cdots a_{n-1} \sqsubseteq C p'_1 \cdots p'_{n-1}$, and $t_2 \to^* p'_n$, $a_n \sqsubseteq p'_n$.
We now focus on the reduction sequence

$$C p'_1 \cdots p'_n \to r^{\vec{p'}} \to^* v$$

Notice that, by the construction sketched above, only redexes that are newly created are contracted, and that any redex created in this sequence corresponds to a redex being created for a sequence starting with $C a_1 \cdots a_n$, therefore

$$C a_1 \cdots a_n \to r^{\vec{a}} \to^* u'$$

and each term created in this reduction is smaller than (in the sense of $\sqsubseteq$) the corresponding term in the reduction sequence above (hence $u' \sqsubseteq v$), and each redex in u' corresponds to a redex in v. Take $a' = \mathcal{DA}_{\mathcal{C}}(v)$, then $a' \sqsubseteq v$, and all redexes are masked by $\bot$. Since $u' \sqsubseteq v$ by masking all the 'old' redexes, we also have that $a' = \mathcal{DA}_{\mathcal{C}}(u')$. Since $a \sqsubseteq u$, also $a \sqsubseteq v$ (by Lemma *6.5(i)*), and therefore $a \sqsubseteq a'$ (by Lemma *6.4(iii)*). We then deduce $a \sqsubseteq u'$. ■

To come to a notion of type assignment on $T(\mathcal{C}, \mathcal{X}, \bot)$, the definition of type assignment as given in Definition *5.7* need *not* be changed, it suffices that the terms are allowed to be in $T(\mathcal{C}, \mathcal{X}, \bot)$. In particular, $\mathcal{E}$ does not produce a type for $\bot$; since $\bot \notin \mathcal{C}$, and because of Definition *5.7*, this implies that $\bot$ can only appear in (sub)terms that are typed with ω.

We will need the following result.

Lemma 6.12 *i) If* $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$, $t \sqsubseteq v$, *then there exists* $\mathrm{D}' :: B \vdash_{\mathcal{E}} v{:}\sigma$, *where the type-derivation* D' *has the same tree-structure as* D *(that is, the same rules are applied).*

ii) If $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$, *and* $t \sqsubseteq v$, *then there exists* $\mathrm{D}' :: B \vdash_{\mathcal{E}} v{:}\sigma$.

Proof: Easy. ■

6.1 Approximation and normalization

In this section we will give the proofs for the approximation and normalisation results.

We will need the following intermediate result.

Lemma 6.13 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS, *then, for all* $t \in \mathcal{T}(\mathcal{C},\mathcal{V})$: *if* $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$ *is in normal form with respect to* $\rightarrow_{\mathcal{D}}$, *then there exists an* $a \in \mathcal{A}_{\mathcal{C}}$ *and* D' *such that* $a \sqsubseteq t$ *and* $\mathrm{D}' :: B \vdash_{\mathcal{E}} a{:}\sigma$.

Proof: By induction on the structure of derivations.

$(\rightarrow E)$: Then $\mathrm{D} = \langle \mathrm{D}_1 :: B \vdash_{\mathcal{E}} t_1{:}\tau\rightarrow\sigma, \mathrm{D}_2 :: B \vdash_{\mathcal{E}} t_2{:}\tau, \rightarrow E\rangle :: B \vdash_{\mathcal{E}} t_1 t_2{:}\sigma$. Then, by induction, there are $a_1 \sqsubseteq t_1, a_2 \sqsubseteq t_2$ such that $\mathrm{D}'_1 :: B \vdash_{\mathcal{E}} a_1{:}\tau\rightarrow\sigma$, and $\mathrm{D}'_2 :: B \vdash_{\mathcal{E}} a_2{:}\tau$, and $\langle \mathrm{D}'_1 :: B \vdash_{\mathcal{E}} a_1{:}\tau\rightarrow\sigma, \mathrm{D}'_2 :: B \vdash_{\mathcal{E}} a_2{:}\tau, \rightarrow E\rangle :: B \vdash_{\mathcal{E}} a_1 a_2{:}\sigma$. By Definition *6.2* we know that $a_1 a_2 \sqsubseteq t_1 t_2$. Now $a_1 a_2 \notin \mathcal{A}_{\mathcal{C}}$ if there is a $C \in \mathcal{C}$ such that $a_1 = C a_1^1 \cdots a_1^{n-1}$ and *arity*$(C) = n$. But then there are $t_1^1, \ldots, t_1^{n-1}$ with $t_1 = C t_1^1 \cdots t_1^{n-1}$, and $t = C t_1^1 \cdots t_1^{n-1} t_2$. In particular, by the remark before Theorem *5.19*, D is reducible, which is impossible. So $a_1 a_2 \in \mathcal{A}_{\mathcal{C}}$.

$(\cap I)$: $\mathrm{D} = \langle \mathrm{D}_1 :: B \vdash_{\mathcal{E}} t{:}\sigma_1, \ldots, \mathrm{D}_n :: B \vdash_{\mathcal{E}} t{:}\sigma_n, \cap I\rangle :: B \vdash_{\mathcal{E}} t{:}\cap_{\underline{n}}\sigma_i$. By induction, for $i \in \underline{n}$, there is an $a_i \sqsubseteq t$ in $\mathcal{A}_{\mathcal{C}}$ such that $\mathrm{D}'_i :: B_i \vdash_{\mathcal{E}} a_i{:}\sigma_i$. Take $a = a_1 \sqcup \cdots \sqcup a_n$. Since, for $i \in \underline{n}$, $a_i \sqsubseteq a$, by Lemma *6.12* also $\mathrm{D}''_i :: B_i \vdash_{\mathcal{E}} a{:}\sigma_i$, so we get

$$\langle \mathrm{D}''_1 :: B_1 \vdash_{\mathcal{E}} a{:}\sigma_1, \ldots, \mathrm{D}''_n :: B_n \vdash_{\mathcal{E}} a{:}\sigma_n, \cap I\rangle :: \underline{B} \vdash_{\mathcal{E}} a{:}\cap_{\underline{n}}\sigma_i.$$

Since $a_i \sqsubseteq t$ for all $i \in \underline{n}$, by Lemma *6.7*, also $a \sqsubseteq t$. Notice that if $n = 0$, then $a = \bot$.

The cases $(\mathcal{E})$ and (*Ax*) are immediate. ■

Theorem 6.14 (APPROXIMATION) *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS, *then: if* $B \vdash_{\mathcal{E}} t{:}\sigma$, *then there exists an* $a \in \mathcal{A}_{\mathcal{C}}(t)$ *such that* $B \vdash_{\mathcal{E}} a{:}\sigma$, *for all* $t \in \mathcal{T}(\mathcal{C},\mathcal{V})$.

Proof: Suppose $\mathrm{D} :: B \vdash_{\mathcal{E}} t{:}\sigma$, then, by Theorem *5.19*, *SN*(D). Let $\mathrm{D}' :: B' \vdash_{\mathcal{E}} v{:}\sigma$ be a normal form of D' with respect to $\rightarrow_{\mathcal{D}}$. Then by Lemma *6.13*, there is an $a \in \mathcal{A}_{\mathcal{C}}$ such that $a \sqsubseteq v$ and $\mathrm{D}'' :: B \vdash_{\mathcal{E}} a{:}\sigma$. Then $t \rightarrow^* v$, therefore $a \in \mathcal{A}_{\mathcal{C}}(t)$. ■

For principal environments we can show that the converse of this result also holds.

Theorem 6.15 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS, *and* $\mathcal{E}$ *be principal for* **C**, *then, if there is an* $a \in \mathcal{A}_{\mathcal{C}}(t)$ *such that* $B \vdash_{\mathcal{E}} a{:}\sigma$, *then* $B \vdash_{\mathcal{E}} t{:}\sigma$.

Proof: If $a \in \mathcal{A}_{\mathcal{C}}(t)$ such that $B \vdash_{\mathcal{E}} a{:}\sigma$, then there exists a v such that $t \rightarrow^* v$ and $a \sqsubseteq v$. But then, by Lemma *6.12*, also $B \vdash_{\mathcal{E}} v{:}\sigma$. Since $\mathcal{E}$ is principal for **C**, by Theorem *5.18*, also $B \vdash_{\mathcal{E}} t{:}\sigma$. ■

Theorem 6.16 (HEAD-NORMALISATION) *Let* $t \in \mathcal{T}(\mathcal{C},\mathcal{V})$. *If* $B \vdash_{\mathcal{E}} t{:}\sigma$, *and* $\sigma \neq \omega$, *then* t *has a head-normal form.*

Proof: If $B \vdash_{\mathcal{E}} t{:}\sigma$, then by Theorem *6.14*, there is an $a \in \mathcal{A}_{\mathcal{C}}(t)$ such that $B \vdash_{\mathcal{E}} a{:}\sigma$. Since $\sigma \neq \omega$, $a \neq \bot$, and since $a \in \mathcal{A}_{\mathcal{C}}$, there are x or C, and terms $a_1, \ldots, a_n$ such that $a = x\,a_1 \cdots a_n$, or $a = C\,a_1 \cdots a_n$ with *arity*$(C) < n$. Also, since $a \in \mathcal{A}_{\mathcal{C}}(t)$, there is a v such that $t \twoheadrightarrow v$ and $a \sqsubseteq v$. Since $a \sqsubseteq v$, there are $t_1, \ldots, t_n$ such that either $v = x\,t_1 \cdots t_n$, or $v = C\,t_1 \cdots t_n$, with *arity*$(C) < n$. But then v *is in* head-normal form, so t *has* a head-normal form. ■

The combinatorial equivalent of another well-known result for intersection type assignment in the LC, i.e. the property

If $B \vdash_{\mathcal{E}} t{:}\sigma$, and B, σ are ω-free, then t has a normal form

no longer holds.

Example 6.17 Take the CS

$$\begin{array}{rcll} \mathsf{Z}xy & \to & y & \qquad \mathcal{E}\,(\mathsf{Z}) = \omega{\to}\varphi_1{\to}\varphi_1, \\ \mathsf{D}x & \to & xx & \qquad \mathcal{E}\,(\mathsf{D}) = ((\varphi_2{\to}\varphi_3){\cap}\varphi_2){\to}\varphi_3 \end{array}$$

then $\mathsf{Z}(\mathsf{DD})$ is typeable with a type not containing ω, but the term $\mathsf{Z}(\mathsf{DD})$ has no normal form.

We will now show that, using Theorem *5.19*, all terms typeable in the subsystem of $\vdash_{\mathcal{E}}$ that does not use ω ($\vdash^{\omega}_{\mathcal{E}}$), are strongly normalizable.

Lemma 6.18 *i) If* D *is a derivation in* $\vdash^{\omega}_{\mathcal{E}}$ *, and* D $\to_{\mathcal{D}}$ D′*, then also* D′ *is a derivation in* $\vdash^{\omega}_{\mathcal{E}}$.

ii) D $:: B \vdash^{\omega}_{\mathcal{E}} t{:}\sigma \to_{\mathcal{D}}$ D′ $:: B' \vdash^{\omega}_{\mathcal{E}} t'{:}\sigma$*, if and only if* $t \to t'$.

Thus, in the type system $\vdash^{\omega}_{\mathcal{E}}$, $\to_{\mathcal{D}}$ mimics $\to$ and vice-versa. This observation immediately leads to the following result.

Theorem 6.19 *Let $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$. If $B \vdash^{\omega}_{\mathcal{E}} t{:}\sigma$, then t is strongly normalizable.*

Proof: Let D be such that D $:: B \vdash^{\omega}_{\mathcal{E}} t{:}\sigma$. Since also D $:: B \vdash_{\mathcal{E}} t{:}\sigma$, by Theorem *5.19*, D is strongly normalizable with respect to $\to_{\mathcal{D}}$. By Lemma *6.18(ii)*, all derivation redexes in D correspond to redexes in t and vice-versa, a property that is preserved under reduction. So also t is strongly normalizable. ■

It is worthwhile to notice that, unlike for LC with $\vdash$, the reverse implication of the three theorems does not hold in general. For this, it is sufficient to note that a subject expansion theorem does not hold (see also the last remark of Section *5.3*).

Another aspect worth noting is that, unlike in LC, no longer every term in normal form is typeable without ω in basis and type. Take for example

$$t = \mathsf{S}(\mathsf{K}(\mathsf{SII}))(\mathsf{K}(\mathsf{SII})),$$

and note that, by Property *5.10* every type assignable to t (regardless of the environment used) is a type assignable to $\lambda y.(\lambda x.xx)(\lambda x.xx)$ in $\vdash$. Since this last term has no head-normal form, only ω can be assigned to it.

7 Semantics

In this section, we will define two semantics for CS. The first is a filter model, where terms will be interpreted by the set of their assignable types; the second an approximation model, where terms will be interpreted by the set of their approximants.

Definition 7.1 Application on $\wp\mathcal{A}_\mathcal{C}$, $\cdot : \wp\mathcal{A}_\mathcal{C} \times \wp\mathcal{A}_\mathcal{C} \to \wp\mathcal{A}_\mathcal{C}$, is defined by:

$$A_1 \cdot A_2 = \{a \in \mathcal{A}_\mathcal{C} \mid \exists a_1 \in A_1, a_2 \in A_2, u\, [a_1 a_2 \to^* u \,\&\, a \sqsubseteq u]\}.$$

We will define two interpretations of terms:

Definition 7.2 *i*) The interpretation of terms in the domain of approximants over $\mathcal{C}$ is defined as: $[\![t]\!]^\mathcal{A}_\mathcal{C} = \mathcal{A}_\mathcal{C}(t) = \{a \in \mathcal{A}_\mathcal{C} \mid \exists u\, [t \to^* u \,\&\, a \sqsubseteq u]\}$.

ii) Let ξ be a valuation of term variables in $\mathcal{F}$; we write $\xi \models B$ if and only if, for all $x{:}\sigma \in B$, $\sigma \in \xi(x)$. $[\![t]\!]^\mathcal{F}_{\xi,\mathcal{E}}$, the interpretation of terms in $\mathcal{F}$ via ξ and $\mathcal{E}$ is defined by: $[\![t]\!]^\mathcal{F}_{\xi,\mathcal{E}} = \{\sigma \mid \exists B\, [\xi \models B \,\&\, B \vdash_\mathcal{E} t{:}\sigma]\}$.

Notice that, by rule $(\cap I)$ and Theorem *5.13*, $\{\sigma \mid \exists B\, [B \vdash_\mathcal{E} t{:}\sigma]\} \in \mathcal{F}$.

Both notions of application, as well as that on sets of approximants as that on filters, are well-defined, in the sense that they respect application on terms.

Lemma 7.3 *i)* $[\![t_1]\!]^\mathcal{A}_\mathcal{C} \cdot [\![t_2]\!]^\mathcal{A}_\mathcal{C} = [\![t_1 t_2]\!]^\mathcal{A}_\mathcal{C}$.

ii) $[\![t_1]\!]^\mathcal{F}_{\xi,\mathcal{E}} \cdot [\![t_2]\!]^\mathcal{F}_{\xi,\mathcal{E}} = [\![t_1 t_2]\!]^\mathcal{F}_{\xi,\mathcal{E}}$.

Proof: *i*) $(\subseteq)$: $[\![t_1]\!]^\mathcal{A}_\mathcal{C} \cdot [\![t_2]\!]^\mathcal{A}_\mathcal{C} =$ (*7.1*)

$\{a \in \mathcal{A}_\mathcal{C} \mid \exists a_1 \in [\![t_1]\!]^\mathcal{A}_\mathcal{C}, a_2 \in [\![t_2]\!]^\mathcal{A}_\mathcal{C}, u[a_1 a_2 \to^* u \,\&\, a \sqsubseteq u]\} =$

$\{a \in \mathcal{A}_\mathcal{C} \mid \exists a_1, a_2 \in \mathcal{A}_\mathcal{C}, u\, [\exists u_1\, [t_1 \to^* u_1 \,\&\, a_1 \sqsubseteq u_1] \,\&$

$\exists u_2\, [t_2 \to^* u_2 \,\&\, a_2 \sqsubseteq u_2] \,\&\, a_1 a_2 \to^* u \,\&\, a \sqsubseteq u]\} \subseteq$ (*6.5(ii)*)

$\{a \in \mathcal{A}_\mathcal{C} \mid \exists u\, [t_1 t_2 \to^* u \,\&\, a \sqsubseteq u]\} =$

$[\![t_1 t_2]\!]^\mathcal{A}_\mathcal{C}$

$(\supseteq)$: $[\![t_1 t_2]\!]^\mathcal{A}_\mathcal{C} = \{a \in \mathcal{A}_\mathcal{C} \mid \exists u\, [t_1 t_2 \to^* u \,\&\, a \sqsubseteq u]\} \subseteq$ (*6.11*)

$\{a \in \mathcal{A}_\mathcal{C} \mid \exists a_1 \in [\![t_1]\!]^\mathcal{A}_\mathcal{C}, a_2 \in [\![t_2]\!]^\mathcal{A}_\mathcal{C}, u[a_1 a_2 \to^* u \,\&\, a \sqsubseteq u]\} =$

$[\![t_1]\!]^\mathcal{A}_\mathcal{C} \cdot [\![t_2]\!]^\mathcal{A}_\mathcal{C}$

ii) $[\![t_1]\!]^\mathcal{F}_{\xi,\mathcal{E}} \cdot [\![t_2]\!]^\mathcal{F}_{\xi,\mathcal{E}} =$

$\uparrow\{\sigma \mid \exists\, \tau \in [\![t_2]\!]^\mathcal{F}_{\xi,\mathcal{E}}\, [\tau{\to}\sigma \in [\![t_1]\!]^\mathcal{F}_{\xi,\mathcal{E}}]\} =$

$\uparrow\{\sigma \mid \exists \tau\, [\exists B_1\, [B_1 \vdash_\mathcal{E} t_1{:}\tau{\to}\sigma] \,\&\, \exists B_2\, [B_2 \vdash_\mathcal{E} t_2{:}\tau]]\} =$ $(\subseteq\colon B = \bigcap\{B_1, B_2\})$

$\uparrow\{\sigma \mid \exists \tau, B\, [B \vdash_\mathcal{E} t_1{:}\tau{\to}\sigma \,\&\, B \vdash_\mathcal{E} t_2{:}\tau]\} =$

$\uparrow\{\sigma \mid \exists B\, [B \vdash_\mathcal{E} t_1 t_2{:}\sigma]\} =$

$\{\sigma \mid \exists B\, [B \vdash_\mathcal{E} t_1 t_2{:}\sigma]\} =$

$[\![t_1 t_2]\!]^\mathcal{F}_{\xi,\mathcal{E}}$ ■

7.1 The relation $=_\mathbf{R}$: equating terms through $\to_\mathbf{R}$

As seen above in Lemma *6.9(i)*, if $t \to^* t'$, then $\mathcal{A}_\mathcal{C}(t) = \mathcal{A}_\mathcal{C}(t')$, which implies that, at least, if $t \to^* t'$, then $[\![t]\!]^\mathcal{A}_\mathcal{C} = [\![t']\!]^\mathcal{A}_\mathcal{C}$. The converse does not hold, since unsolvable terms that are not

in $\rightarrow^*$, still have the same image under $[\![\ ]\!]^{\mathcal{A}}_{\mathcal{C}}$, namely $\bot$. We now formalize these properties.

The relation $=_{\mathbf{R}}$ is the reflexive, symmetric and transitive closure of $\rightarrow_{\mathbf{R}}$:

Definition 7.4 Let $((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS. We define the equivalence relation $=_{\mathbf{R}} \subseteq \mathcal{T}(\mathcal{C},\mathcal{V}) \times \mathcal{T}(\mathcal{C},\mathcal{V})$ by:

$$\begin{array}{rcl} t \rightarrow^*_{\mathbf{R}} v & \Rightarrow & t =_{\mathbf{R}} v \\ t =_{\mathbf{R}} v & \Rightarrow & v =_{\mathbf{R}} t \\ t =_{\mathbf{R}} v \ \& \ v =_{\mathbf{R}} w & \Rightarrow & t =_{\mathbf{R}} w \end{array}$$

Lemma 7.5 *If $t =_{\mathbf{R}} v$, then there exists u such that $t \rightarrow^*_{\mathbf{R}} u$ and $v \rightarrow^*_{\mathbf{R}} u$.*

Proof: By induction on the definition of $=_{\mathbf{R}}$. If $t =_{\mathbf{R}} v \ \& \ v =_{\mathbf{R}} w \ \Rightarrow \ t =_{\mathbf{R}} w$, then, by induction there are u_1 and u_2 such that $t \rightarrow^*_{\mathbf{R}} u_1$ and $v \rightarrow^*_{\mathbf{R}} u_1$, and $v \rightarrow^*_{\mathbf{R}} u_2$ and $w \rightarrow^*_{\mathbf{R}} u_2$. Since $v \rightarrow^*_{\mathbf{R}} u_1$ and $v \rightarrow^*_{\mathbf{R}} u_2$, by Property *4.5*, there exist a u_3 such that $u_1 \rightarrow^*_{\mathbf{R}} u_3$ and $u_2 \rightarrow^*_{\mathbf{R}} u_3$. But then, in particular, $t \rightarrow^*_{\mathbf{R}} u_3$ and $w \rightarrow^*_{\mathbf{R}} u_3$. The other cases are straightforward. ■

The approximant semantics is adequate, in that it equates terms that are equal in the theory **R**.

Theorem 7.6 (ADEQUACY OF THE APPROXIMATION MODEL) *If $t =_{\mathbf{R}} v$, then $[\![t]\!]^{\mathcal{A}}_{\mathcal{C}} = [\![v]\!]^{\mathcal{A}}_{\mathcal{C}}$.*

Proof: Consequence of Lemma *7.5* and *6.9(i)*. ■

The converse of this result, '*If $[\![t]\!]^{\mathcal{A}}_{\mathcal{C}} = [\![v]\!]^{\mathcal{A}}_{\mathcal{C}}$, then $t =_{\mathbf{R}} v$*' does not hold.

Example 7.7 Take the CS

$$\begin{array}{rcl} \mathsf{S}xyz & \rightarrow & xz(yz) \\ \mathsf{K}xy & \rightarrow & x \\ \mathsf{D}x & \rightarrow & xx \\ \mathsf{W}x & \rightarrow & xxx \end{array}$$

Notice that $\mathsf{SK}(\mathsf{DD})$ and $\mathsf{SK}(\mathsf{WW})$ both have only one redex, and that this property is preserved under reduction. Then

$$\mathsf{SK}(\mathsf{DD}) \rightarrow \mathsf{SK}(\mathsf{DD}) \rightarrow \mathsf{SK}(\mathsf{DD}) \rightarrow \cdots$$

and

$$\mathsf{SK}(\mathsf{WW}) \rightarrow \mathsf{SK}(\mathsf{WWW}) \rightarrow \mathsf{SK}(\mathsf{WWWW}) \rightarrow \cdots,$$

so

$$[\![\mathsf{SK}(\mathsf{DD})]\!]^{\mathcal{A}}_{\mathcal{C}} = \{\bot, \mathsf{S}\bot\bot, \mathsf{SK}\bot\} = [\![\mathsf{SK}(\mathsf{WW})]\!]^{\mathcal{A}}_{\mathcal{C}},$$

but there is no u such that both $\mathsf{SK}(\mathsf{DD}) \rightarrow^* u$ and $\mathsf{SK}(\mathsf{WW}) \rightarrow^* u$.

7.2 The relation $\approx_{\mathbf{R}}$: $=_{\mathbf{R}}$ and equating unsolvables

We could modify the relation $=_{\mathbf{R}}$ to identify all unsolvable terms, so $\mathsf{SK}(\mathsf{DD}) \approx_{\mathbf{R}} \mathsf{SK}(\mathsf{WW})$ (this is used also for LC).

Definition 7.8 Let $((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS. We define the equivalence relation $\approx_{\mathbf{R}} \subseteq \mathcal{T}(\mathcal{C},\mathcal{V}) \times \mathcal{T}(\mathcal{C},\mathcal{V})$ by:

$$\begin{array}{rcl} t \to_{\mathbf{R}}^{*} v & \Rightarrow & t \approx_{\mathbf{R}} v \\ t, v \text{ are unsolvable} & \Rightarrow & t \approx_{\mathbf{R}} v \\ t \approx_{\mathbf{R}} v & \Rightarrow & v \approx_{\mathbf{R}} t \\ t \approx_{\mathbf{R}} v \;\&\; v \approx_{\mathbf{R}} w & \Rightarrow & t \approx_{\mathbf{R}} w \\ t \approx_{\mathbf{R}} v & \Rightarrow & wt \approx_{\mathbf{R}} wv \;\&\; tw \approx_{\mathbf{R}} vw \end{array}$$

Notice that $\mathsf{SK}(\mathsf{DD}) \approx_{\mathbf{R}} \mathsf{SK}(\mathsf{WW})$.

Theorem 7.9 *If $t \approx_{\mathbf{R}} v$, then $[\![t]\!]_{\mathcal{C}}^{\mathcal{A}} = [\![v]\!]_{\mathcal{C}}^{\mathcal{A}}$.*

Proof: By induction on the definition of $\approx_{\mathbf{R}}$. The case $t \to_{\mathbf{R}}^{*} v$ follows from Lemma *6.9(i)*. If t, v are unsolvable, then $[\![t]\!]_{\mathcal{C}}^{\mathcal{A}} = \{\bot\} = [\![v]\!]_{\mathcal{C}}^{\mathcal{A}}$. The last case is a consequence of Lemma *7.3*. The other two cases follow by induction. ■

Although, by $\approx_{\mathbf{R}}$, terms are equated that are unsolvable, still we do not get a full-abstraction result, since it can be that solvable terms have the same infinite set of approximants, whilst sharing no terms during reduction.

Example 7.10 Take

$$\begin{array}{rcl} \mathsf{T}xy & \to & y(xxy) \\ \mathsf{Y}xy & \to & y(xy(xy)) \\ \mathsf{X}xy & \to & x(yy) \end{array}$$

Then we have the following reduction sequences:

$$\begin{array}{rcl} \mathsf{YX}z & \to & z(\mathsf{X}z(\mathsf{X}z)) \\ & \to & z(z(\mathsf{X}z(\mathsf{X}z))) \\ & \to & z(z(z(\mathsf{X}z(\mathsf{X}z)))) \\ & \cdots & \\ & \to & z(z(z(z(z\cdots)))) \end{array} \qquad \begin{array}{rcl} \mathsf{TT}z & \to & z(\mathsf{TT}z) \\ & \to & z(z(\mathsf{TT}z)) \\ & \to & z(z(z(\mathsf{TT}z))) \\ & \cdots & \\ & \to & z(z(z(z(z\cdots)))) \end{array}$$

In particular,

$$[\![\mathsf{YX}z]\!]_{\mathcal{C}}^{\mathcal{A}} = \{\bot, z\bot, z(z\bot), z(z(z\bot)), \ldots\} = [\![\mathsf{TT}z]\!]_{\mathcal{C}}^{\mathcal{A}},$$

but *not* $\mathsf{YX}z \approx_{\mathbf{R}} \mathsf{TT}z$.

7.3 The relation $\approx_{\mathbf{R}}^{hnf}$: full-abstraction

We can obtain a full-abstraction result for the approximation semantics using the following relation:

Definition 7.11 Let $((\mathcal{C},\mathcal{X}),\mathbf{R})$ be a CS. The relation $\approx_{\mathbf{R}}^{hnf}$ is defined co-inductively as follows: $t \approx_{\mathbf{R}}^{hnf} u$ if and only if either

i) t and u are both unsolvable, or

ii) if $Ct_1 \cdots t_n$ is a head normal form of t (resp. u), then there is a head normal form $Cu_1 \cdots u_n$ of u (resp. t) such that, for $i \in \underline{n}$, $t_i \approx_{\mathbf{R}}^{hnf} u_i$, or

iii) if $x t_1 \cdots t_n$ is a head normal form of t (resp. u), then there is a head normal form $x u_1 \cdots u_n$ of u (resp. t) such that, for $i \in \underline{n}$, $t_i \approx_{\mathbf{R}}^{hnf} u_i$.

Theorem 7.12 (FULL ABSTRACTION OF THE APPROXIMATION MODEL) *$t \approx_{\mathbf{R}}^{hnf} u$ if and only if $[\![t]\!]_{\mathcal{C}}^{\mathcal{A}} = [\![u]\!]_{\mathcal{C}}^{\mathcal{A}}$.*

Proof: (*if*) : By co-induction. It is sufficient to show that if $[\![t]\!]_{\mathcal{C}}^{\mathcal{A}} = [\![u]\!]_{\mathcal{C}}^{\mathcal{A}}$ then either

a) t, u are unsolvable, or

b) if $C t_1 \cdots t_n$ is a head normal form of t (resp. u), then $C u_1 \cdots u_n$ is a head normal form of u (resp. t), and $[\![t_i]\!]_{\mathcal{C}}^{\mathcal{A}} = [\![u_i]\!]_{\mathcal{C}}^{\mathcal{A}}$ for $i \in \underline{n}$, or

c) if $x t_1 \cdots t_n$ is a head normal form of t (resp. u), then $x u_1 \cdots u_n$ is a head normal form of u (resp. t), and $[\![t_i]\!]_{\mathcal{C}}^{\mathcal{A}} = [\![u_i]\!]_{\mathcal{C}}^{\mathcal{A}}$ for $i \in \underline{n}$.

This is a straightforward consequence of the fact that u and t have the same set of approximants.

(*only if*) : We take $a \in [\![t]\!]_{\mathcal{C}}^{\mathcal{A}}$ and show $a \in [\![u]\!]_{\mathcal{C}}^{\mathcal{A}}$ by induction on the depth of a.

$(a = \bot)$: Trivial.

$(a = C a_1 \ldots a_n)$: Then t has a head normal form $C t_1 \cdots t_n$, and therefore u has a head normal form $C u_1 \cdots u_n$ such that $t_i \approx_{\mathbf{R}}^{hnf} u_i$ for $i \in \underline{n}$. Since $a_i \in [\![t_i]\!]_{\mathcal{C}}^{\mathcal{A}}$ and its depth is smaller than that of a, by induction we conclude that $a_i \in [\![u_i]\!]_{\mathcal{C}}^{\mathcal{A}}$. Therefore $a \in [\![u]\!]_{\mathcal{C}}^{\mathcal{A}}$.

$(a = x a_1 \ldots a_n)$: Similar. ■

7.4 Filter semantics and full abstraction

The filter semantics gives a semi-model with respect to $\rightarrow_{\mathbf{R}}$, as the following theorem shows.

Theorem 7.13 *If $t \rightarrow_{\mathbf{R}}^{*} v$, then $[\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}} \subseteq [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$.*

Proof: Take $\sigma \in [\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$. Then there exists B such that $B \vdash_{\mathcal{E}} t{:}\sigma$, and, since $t \rightarrow_{\mathbf{R}}^{*} v$, by Theorem *5.17*, also $B \vdash_{\mathcal{E}} v{:}\sigma$, so $\sigma \in [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$. ■

In view of the fact that type assignment in $\vdash_{\mathcal{E}}$ is not closed for subject-expansion (see the remark at the end of Section *5.3*), it is, in general, not possible to show a stronger result like '*If $t =_{\mathbf{R}} v$, then $[\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}} = [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$*'. However, when using a principal environment, this result holds.

Theorem 7.14 (ADEQUACY OF THE FILTER MODEL) *Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a* CS*, and $\mathcal{E}$ be principal for $\mathbf{C}$, then $t =_{\mathbf{R}} v$ implies $[\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}} = [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$.*

Proof: By Theorem *5.17* and *5.18*. ■

We even have the following result easily.

Theorem 7.15 *Let $\mathbf{C} = ((\mathcal{C}, \mathcal{X}), \mathbf{R})$ be a* CS*, and $\mathcal{E}$ be principal for $\mathbf{C}$, then $t \approx_{\mathbf{R}} v$ implies $[\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}} = [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$, for all $t, v \in \mathcal{T}(\mathcal{C}, \mathcal{V})$.*

Proof: By induction on the definition of $\approx_{\mathbf{R}}$. The case $t \rightarrow_{\mathbf{R}}^{*} v$ is covered by Theorem *5.17* and *5.18*. If t, v are unsolvable, then by Theorem *6.16*, $[\![t]\!]_{\xi,\mathcal{E}}^{\mathcal{F}} = \{\omega\} = [\![v]\!]_{\xi,\mathcal{E}}^{\mathcal{F}}$. The last case

is a consequence of Lemma *7.3*. The other two cases follow by straightforward induction. ■

The converse of these results do not hold.

Example 7.16 Take $\mathsf{T}, \mathsf{Y}, \mathsf{X}$ as in Example *7.10*, and let

$$\begin{aligned}\mathcal{E}(\mathsf{T}) &= ((\varphi_1{\to}\varphi_2{\to}\varphi_3){\cap}\varphi_1){\to}((\varphi_3{\to}\varphi_4){\cap}\varphi_2){\to}\varphi_4\\ \mathcal{E}(\mathsf{Y}) &= ((\varphi_3{\to}\varphi_5{\to}\varphi_1){\cap}(\varphi_4{\to}\varphi_5)){\to}((\varphi_1{\to}\varphi_2){\cap}\varphi_3{\cap}\varphi_4){\to}\varphi_2\\ \mathcal{E}(\mathsf{X}) &= (\varphi_1{\to}\varphi_2){\to}((\varphi_3{\to}\varphi_1){\cap}\varphi_3){\to}\varphi_2\end{aligned}$$

then

$$\begin{aligned}\llbracket\mathsf{YX}\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}} = \{&\omega, (\omega{\to}\varphi_1){\to}\varphi_1, ((\omega{\to}\varphi_1){\cap}(\varphi_1{\to}\varphi_2)){\to}\varphi_2,\\ &((\omega{\to}\varphi_1){\cap}(\varphi_1{\to}\varphi_2){\cap}(\varphi_2{\to}\varphi_3)){\to}\varphi_3, \ldots\} = \llbracket\mathsf{TT}\rrbracket^{\mathcal{F}}\end{aligned}$$

(notice that these types correspond directly to the approximants of Example *7.10*) but neither $\mathsf{YX} =_{\mathbf{R}} \mathsf{TT}$, nor $\mathsf{YX} \approx_{\mathbf{R}} \mathsf{TT}$.

For the filter semantics, we have, as can be expected:

Theorem 7.17 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS*, and* $\mathcal{E}$ *be principal for* $\mathbf{C}$*, then* $t \approx^{hnf}_{\mathbf{R}} u$ *implies* $\llbracket t\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}} = \llbracket u\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$*, for all* $t, u \in \mathcal{T}(\mathcal{C},\mathcal{V})$.

Proof: If $t \approx^{hnf}_{\mathbf{R}} u$, then, by Theorem *7.12*, $\llbracket t\rrbracket^{\mathcal{A}}_{\mathcal{C}} = \llbracket u\rrbracket^{\mathcal{A}}_{\mathcal{C}}$. Let $\sigma \in \llbracket t\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$ (the other case is similar), then there exists a B such that $B \vdash_{\mathcal{E}} t{:}\sigma$. Then, by Theorem *6.14*, there exists an $a \in \mathcal{A}_{\mathcal{C}}(t)$ such that $B \vdash_{\mathcal{E}} a{:}\sigma$. Since $\mathcal{A}_{\mathcal{C}}(t) = \llbracket t\rrbracket^{\mathcal{A}}_{\mathcal{C}} = \llbracket u\rrbracket^{\mathcal{A}}_{\mathcal{C}} = \mathcal{A}_{\mathcal{C}}(u)$, $a \in \mathcal{A}_{\mathcal{C}}(u)$, and by Theorem *6.15*, $B \vdash_{\mathcal{E}} u{:}\sigma$, so $\sigma \in \llbracket u\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$. ■

Perhaps surprisingly (at least for LC, the approximation and the filter semantics coincide [42, 3]), we do not have a full-abstraction result with respect to filter semantics.

Example 7.18 Take

$$\begin{aligned}\mathsf{E}xy &\to xy\\ \mathsf{I}x &\to x\end{aligned} \qquad \text{and} \qquad \begin{aligned}\mathcal{E}(\mathsf{E}) &= (\varphi_1{\to}\varphi_2){\to}\varphi_1{\to}\varphi_2\\ \mathcal{E}(\mathsf{I}) &= \varphi_1{\to}\varphi_1\end{aligned}$$

Then $\llbracket\mathsf{EI}\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}} = \llbracket\mathsf{I}\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$ but neither $\mathsf{EI} =_{\mathbf{R}} \mathsf{I}$, nor $\mathsf{EI} \approx_{\mathbf{R}} \mathsf{I}$, nor $\mathsf{EI} \approx^{hnf}_{\mathbf{R}} \mathsf{I}$.

The relation between the two semantics is formulated by:

Theorem 7.19 $\llbracket t\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}} \subseteq \bigcup_{a \in \llbracket t\rrbracket^{\mathcal{A}}_{\mathcal{C}}} \llbracket a\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$.

Proof: If $\sigma \in \llbracket t\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$, then there is a B such that $B \vdash_{\mathcal{E}} t{:}\sigma$. Then, by Theorem *6.14*, there is an $a \in \mathcal{A}_{\mathcal{C}}(t)$ such that $B \vdash_{\mathcal{E}} a{:}\sigma$. ■

Note that the inclusion is strict, since the Subject Expansion property does not hold in general. Also, as can be expected:

Theorem 7.20 *Let* $\mathbf{C} = ((\mathcal{C},\mathcal{X}),\mathbf{R})$ *be a* CS*,* $\mathcal{E}$ *principal for* $\mathbf{C}$*, then* $\bigcup_{a \in \llbracket t\rrbracket^{\mathcal{A}}_{\mathcal{C}}} \llbracket a\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}} \subseteq \llbracket t\rrbracket^{\mathcal{F}}_{\xi,\mathcal{E}}$*, for all* $t \in \mathcal{T}(\mathcal{C},\mathcal{V})$.

Proof: If $\sigma \in \bigcup_{a \in [\![t]\!]^{\mathcal{A}}_{\mathcal{C}}} [\![a]\!]^{\mathcal{F}}_{\xi,\mathcal{E}}$, then there exists $a \in [\![t]\!]^{\mathcal{A}}_{\mathcal{C}}$, B such that $B \vdash_{\mathcal{E}} a{:}\sigma$. Then, by Theorem *6.15*, also $B \vdash_{\mathcal{E}} t{:}\sigma$, so $\sigma \in [\![t]\!]^{\mathcal{F}}_{\xi,\mathcal{E}}$. ■

References

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

[2] S. van Bakel. Principal type schemes for the Strict Type Assignment System. *Logic and Computation*, 3(6):643–670, 1993.

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

[4] S. van Bakel. Rank 2 Intersection Type Assignment in Term Rewriting Systems. *Fundamenta Informaticae*, 26(2):141–166, 1996.

[5] S. van Bakel. *Strongly Normalisation Cut-Elimination with Strict Intersection Types (Extended Abstract)*, International Workshop *Intersection Types and Related Systems 2002* (ITRS'02), Electronic Notes in Theoretical Computer Science, volume 70-1, 2002.

[6] S. van Bakel, F. Barbanera, M. Dezani-Ciancaglini and F.J. de Vries. *Intersection Types for Trees*. *Theoretical Computer Science*, 272 (Theories of Types and Proofs 1997): 3-40, 2002.

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

[8] S. van Bakel and M. Fernández. Normalization Results for Typeable Rewrite Systems. *Information and Computation*, 133(2):73–116, 1997.

[9] S. van Bakel and M. Fernández. *Normalization, Approximation and Semantics for Combinator Systems*. *Theoretical Computer Science*, 290:975-1019, 2003.

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

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

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

[13] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Principal type schemes and λ-calculus semantics. In J.R. Hindley and J.P. Seldin, editors, *To H.B. Curry, Essays in combinatory logic, lambda-calculus and formalism*, pages 535–560. Academic press, New York, 1980.

[14] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 27:45–58, 1981.

[15] M. Coppo, M. Dezani-Ciancaglini, and M. Zacchi. Type Theories, Normal Forms and D_∞-Lambda-Models. *Information and Computation*, 72(2):85–116, 1987.

[16] H.B. Curry. Grundlagen der Kombinatorischen Logik. *American Journal of Mathematics*, 52:509–536, 789–834, 1930.

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

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

[19] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 6, pages 245–320. North-Holland, 1990.

[20] M. Dezani-Ciancaglini and I. Margaria. A characterisation of F-complete type assignments. *Theoretical Computer Science*, 45:121–157, 1986.

[21] M. Dezani-Ciancaglini and J.R. Hindley. Intersection types for combinatory logic. *Theoretical Computer Science*, 100:303–324, 1992.

[22] E. Engeler. Algebras and combinators. *Algebra universalis*, 13(3):389–392, 1981.

[23] K. Futatsugi, J. Goguen, J.P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proceedings 12$^{\text{th}}$ ACM Symposium on Principles of Programming Languages*, pages 52–66, 1985.

[24] J.Y. Girard. The System F of Variable Types, Fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[25] C.A. Gunter and D.S. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 633–674. North-Holland, 1990.

[26] J.R. Hindley. The principal type scheme of an object in combinatory logic. *Transactions of the American Mathematical Society*, 146:29–60, 1969.

[27] J.R. Hindley. The simple semantics for Coppo-Dezani-Sallé type assignment. In M. Dezani and U. Montanari, editors, *International symposium on programming*, volume 137 of *Lecture Notes in Computer Science*, pages 212–226. Springer-Verlag, 1982.

[28] J.R. Hindley. The Completeness Theorem for Typing λ-terms. *Theoretical Computer Science*, 22(1):1–17, 1983.

[29] R. Hindley and G. Longo. Lambda calculus models and extensionality. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 26:289–310, 1980.

[30] G. Huet and J.J. Lévy. Computations in Orthogonal Rewriting Systems. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic. Essays in Honour of Alan Robinson*. MIT Press, 1991.

[31] B. Jacobs, I. Margaria, and M. Zacchi. Filter Models with Polymorphic Types. *Theoretical Computer Science*, 95:143–158, 1992.

[32] J.W. Klop. Term Rewriting Systems. In S. Abramsky, Dov.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1–116. Clarendon Press, 1992.

[33] J.W. Klop and A. Middeldorp. Sequentiality in Orthogonal Term Rewriting Systems. *Journal of Symbolic Computation*, 12:161–195, 1991.

[34] D. Leivant. Polymorphic Type Inference. In *Proceedings 10$^{\text{th}}$ ACM Symposium on Principles of Programming Languages*, Austin Texas, pages 88–98, 1983.

[35] I. Margaria and M. Zacchi. Principal Typing in a $\forall\cap$-Discipline. *Logic and Computation*. To appear.

[36] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

[37] J.C. Mitchell. Polymorphic Type Inference and Containment. *Information and Computation*, 76:211–249, 1988.

[38] B.C. Pierce. *Programming with Intersection Types and Bounded Polymorphism*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pitssburgh, 1991. CMU-CS-91-205.

[39] B.C. Pierce. Intersection Types and Bounded Polymorphism. In M. Bezem and J.F. Groote, editors, *Proceedings of TLCA '93. International Conference on Typed Lambda Calculi and Applications*, Utrecht, the Netherlands, volume 664 of *Lecture Notes in Computer Science*, pages 346–360. Springer-Verlag, 1993.

[40] J.C. Reynolds. The essence of Algol. In J.W. de Bakker and J.C. van Vliet, editors, *Algorithmic languages*, pages 345–372. North-Holland, 1981.

[41] J.C. Reynolds. Preliminary design of the programming language Forsythe. Technical Report CMU-CS-88-159, Carnegie Mellon University, Pitssburgh, 1988.

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

[43] P. Sallé. Une extension de la théorie des types. In G. Ausiello and C. Böhm, editors, *Automata, languages and programming. Fifth Colloquium*, Udine, Italy, volume 62 of *Lecture Notes in Computer Science*, pages 398–410, Udine, Italy, 1978. Springer-Verlag.

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

[45] S.R. Thatte. Full Abstraction and Limiting Completeness in Equational Languages. *Theoretical Computer Science*, 65:85–119, 1989.

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