Intersection Type Assignment Systems

Steffen van Bakel

Afdeling Informatica, Universiteit Nijmegen,
Toernooiveld 1, 6525 ED Nijmegen, Nederland.
steffen@cs.kun.nl*

Abstract

This paper gives an overview of intersection type assignment for the Lambda Calculus, as well as compare in detail variants that have been defined in the past. It presents the essential intersection type assignment system, that will prove to be as powerful as the well-known BCD-system. It is essential in the following sense: it is an almost syntax directed system that satisfies all major properties of the BCD-system, and the types used are the representatives of equivalence classes of types in the BCD-system. The set of typeable terms can be characterized in the same way, the system is complete with respect to the simple type semantics, and it has the principal type property.

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 [14]. 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, is there a basis B and a type σ such that $B \vdash M{:}\sigma$?

is decidable, and that it has the principal type property. These two properties found their way into programming, mainly through the pioneering work of R. Milner [28]. 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 [10] by M. Coppo, M. Dezani-Ciancaglini, and B. Venneri (a more enhanced system was presented in [6] 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

*Supported by the Netherlands Organisation for the advancement of pure research (N.W.O.). Current affiliation: *Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italia, bakel@di.unito.it* 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 [6] (the BCD-system) is based on the system as presented in [10]. It defines the set of intersection types $\mathcal{T}_\cap$ 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 [6] 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 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 [2], as first suggested by D. Leivant in [26], 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 $\sigma_1, \ldots, \sigma_n$ are types that do not contain intersections.

This Rank 2 system was later used as a basis for the notion of type assignment as studied by M. Coppo and P. Giannini in [12]. In that paper the idea behind the let-construct of ML is generalized. In ML, quantification of type-variables is introduced, with the normal restriction that a term variable can only be bound if it does not occur free in the basis (context) for a term; only those type-variables can be instantiated by types. Similarly in [12], only those type-variables can be expanded (for the notion of expansion, see Subsection *5.1*). Since the operation of expansion uses also the notion of quantification, intersection types in this system are limited to those that have the same shape, instead of allowing for arbitrary intersections. The system studied by F. Damiani and P. Giannini in [15] is a restriction of the system in [12], in that an ($\rightarrow$I)-step can only be performed against the types actually used for the term-variable; the notion of type assignment of that paper is, therefore, called *relevant*.

That intersection types can be used as a basis for programming languages was first discussed by J. Reynolds in [32]. This led to the development of the (typed) programming language Forsythe [33], and to the work of B.C. Pierce [30, 31], 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 [6] the relation $\leq$ induced an equivalence relation $\sim$ on types. Equivalence classes are big (for example: $\omega \sim (\sigma \rightarrow \omega)$, for all types σ) and type assignment is closed for $\sim$.

The BCD-system has the principal type property, as was shown in [34]; the set of operations needed for this system consists of substitutions, expansions, and rises; although for every M the set $\{\langle B, \sigma \rangle \mid B \vdash M{:}\sigma\}$ can be generated using those operations specified in [34], 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. 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 Essential Type Assignment System as presented in this paper 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 [6]; this result can also be obtained for the essential system. The main advantage of the essential system over the BCD-system is that the set of types assignable to a term is significantly smaller. An other advantage of the essential 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 essential system will prove to be closed for η-reduction: If $B \vdash_{\mathrm{E}} M{:}\sigma$ and $M \rightarrow_{\eta} N$, then $B \vdash_{\mathrm{E}} N{:}\sigma$. This does not hold for the strict system.

This paper also gives an overview of various notions of intersection type assignment, in order to give insight and intuition, and to compare in detail the various systems and their development. The main objective of this paper is to show that the treatment of intersection types as in [6] has been too general; the same results could have been obtained for a far less complicated system, that follows more closely the syntactical structure of terms, and treats the type ω not as a type constant, but as the empty intersection. Great advantages of this approach are a less complicated type language, less complicated proofs, and more precise and elegant definitions. For example, the operations that are defined in this paper, needed in a proof for the principal type property are 'orthogonal': there is no overlap.

Also, some results already known for, for example, the BCD-system or one of the CDV-systems will be shown to hold for the essential system ($\vdash_{\mathrm{E}}$).

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

In general, this will be done using different techniques like, for example, for Theorems *2.8*, *2.10*, *3.2.7*, and *3.3.2*; the proof of Theorem *6.3.3* uses a standard technique, but is based on new ideas.

The outline of this paper is as follows. Section *1* will give an overview of, in total, five notions of intersection type assignment, that will be compared and discussed. Subsection *1.1* presents the Coppo-Dezani system as defined in [7], Subsection *1.2* presents two Coppo-Dezani-Venneri systems as first defined in [10], Subsection *1.3* presents the Barendregt-Coppo-Dezani system that can be found in [6], and Subsection *1.4* presents the strict system of [1].

Section *2* contains the presentation of the essential type assignment system, an extension of the strict system, and a restriction of the BCD-system that will prove to be equally powerful as the BCD-system. In that subsection it will be shown that type assignment in this system is closed for β-equality (Theorem *2.10*) and η-reduction (Theorem *2.8*). In Section *3* an approximation theorem will be proved: $B \vdash_{\mathrm{E}} M{:}\sigma \iff \exists\, A \in \mathcal{A}(M)\ [B \vdash_{\mathrm{E}} A{:}\sigma]$ (Theorem *3.2.7*), using a notion of computability. With this result, a characterization of the sets of terms having a (head) normal form will be given (Theorem *3.3.2*). In Section *4*, soundness and completeness of essential type assignment will be proved (respectively in Theorem *4.3.1* and Theorem *4.3.6*). The section starts with formulating the relation between the BCD- and the essential system in Subsection *4.1*. In Subsection *4.2* a filter λ-model will be defined, that is used in Subsection *4.3* to prove completeness of type assignment with respect to the simple type semantics (see Definition *1.3.5*).

For three of these systems presented in the past (one CDV system, the BCD system and the strict system) the principal type property has been shown to hold in, respectively, [9], [34], and [4]; in Section *5* the constructions of the proofs of the various papers will be discussed. Finally, in Section *6*, the proof for the principal type property for the essential system will be sketched.

The results presented in this paper first appeared, in a condensed and slightly different form, in [3].

Notations

In this paper, 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. '$\rightarrow$' will be assumed to associate to the right, and '$\cap$' binds stronger than '$\rightarrow$'. M, N are used for lambda terms, x, y, z for term-variables, $M[N/x]$ for the usual operation of substitution in terms, and A for terms in $\lambda\bot$-normal form. 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.

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 A historical perspective

Type assignment for the Lambda Calculus was first studied by H.B. Curry in [13]. (See also [14].) Curry's system – the first and most primitive one – expresses abstraction and application and its major advantage is that the problem of type assignment is decidable. The types used in this system are those obtained from type-variables and the type-constructor $\rightarrow$ '*arrow*'.

Definition 1.1 *i) The set of Curry-types is inductively defined by:*

a) All type-variables $\varphi_0, \varphi_1, \ldots$ are Curry-types.

b) If σ and τ are Curry-types, then so is $\sigma{\rightarrow}\tau$.

ii) Curry-type assignment and Curry-derivations are defined by the following natural deduction system.

$$(\rightarrow I): \quad \frac{\begin{array}{c}[x{:}\sigma]\\ \vdots \\ M{:}\tau\end{array}}{\lambda x.M{:}\sigma{\rightarrow}\tau}\ (a) \qquad\qquad (\rightarrow E): \quad \frac{M{:}\sigma{\rightarrow}\tau \qquad N{:}\sigma}{MN{:}\tau}$$

(a) If $x{:}\sigma$ is the only statement about x on which $M{:}\tau$ depends.

The main results proved for this system are:

- The principal type property: for every typeable M there is a pair $\langle P, \pi\rangle$, such that: $P \vdash M{:}\pi$, and for every pair $\langle B, \sigma\rangle$ such that $B \vdash M{:}\sigma$, there exists a substitution *Sub* such that *Sub* $(\langle P, \pi\rangle) = \langle B, \sigma\rangle$.
- Decidability of type assignment.
- Strongly normalizability of all typeable terms.

Curry's system has drawbacks: it is for example not possible to assign a type to the lambda term $(\lambda x.xx)$, and although the lambda terms $(\lambda cd.d)$ and $((\lambda xyz.xz(yz))(\lambda ab.a))$ are β-equal, the principal type schemes for these terms are different, $\varphi_0{\rightarrow}\varphi_1{\rightarrow}\varphi_1$ and $(\varphi_1{\rightarrow}\varphi_0){\rightarrow}\varphi_1{\rightarrow}\varphi_1$, respectively. The Intersection Type Discipline as presented in the following subsections is an extension of Curry's system for the pure Lambda Calculus that does not have these drawbacks. It has developed over a period of several years; in the sequel, not just the final version will be presented, but various systems that appeared between 1980 and 1992 will be shown, in order to develop a concise overview of the field.

1.1 The Coppo-Dezani type assignment system

The first paper by M. Coppo and M. Dezani-Ciancaglini from the University of Turin, Italy that introduced intersection types is [7] (in this paper, instead of the word 'intersection', the word 'sequence' was used). The system presented in this paper is a true extension of Curry's system: the extension made is to allow of more than one type for term-variables in the ($\rightarrow$I)-derivation rule and therefore to allow of, also, more than one type for the right hand term in the ($\rightarrow$E)-derivation rule.

Definition 1.1.1 *i) The set of types considered in [7] is inductively defined by:*

a) All type-variables $\varphi_0, \varphi_1, \ldots$ are types.

b) If $\sigma_1, \ldots, \sigma_n$ are types ($n \geq 1$), then $\sigma_1 \cap \cdots \cap \sigma_n$ is a sequence*.*

c) If $\sigma_1 \cap \cdots \cap \sigma_n$ is a sequence and σ is a type, then $\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma$ is a type.

ii) Type assignment and derivations are defined by the following natural deduction system.

$$(\cap I): \quad \frac{M{:}\sigma_1 \quad \cdots \quad M{:}\sigma_n}{M{:}\sigma_1 \cap \cdots \cap \sigma_n} \qquad (\rightarrow I): \quad \frac{\begin{matrix}[x{:}\sigma_1 \cap \cdots \cap \sigma_n] \\ \vdots \\ M{:}\sigma\end{matrix}}{\lambda x.M{:}\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma} \; (a)$$

$$(\cap E): \quad \frac{x{:}\sigma_1 \cap \cdots \cap \sigma_n}{x{:}\sigma_i} \qquad (\rightarrow E): \quad \frac{M{:}\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma \qquad N{:}\sigma_1 \cap \cdots \cap \sigma_n}{MN{:}\sigma}$$

(a) If $x{:}\sigma_1 \cap \cdots \cap \sigma_n$ is the only statement about x on which $M{:}\tau$ depends.

The main properties of that system proved in [7] are:

- Subject reduction: If $B \vdash M{:}\sigma$ and $M \rightarrow_\beta N$, then $B \vdash N{:}\sigma$.
- Normalizability of typeable terms: If $B \vdash M{:}\sigma$, then M has a normal form.
- Typeability of all terms in normal form.
- Closure for β-equality in the λI-calculus: if $B \vdash M{:}\sigma$ and $M =_\beta N$, then $B \vdash N{:}\sigma$.
- In the λI-calculus: $B \vdash M{:}\sigma$ if and only if M has a normal form.

It is even possible to prove that, in this system: If $B \vdash M{:}\sigma$, then M is strongly normalizable (in fact, this follows from a similar result of [1]; there the BCD-system without ω is studied, which is a supersystem of the CD-system).

That the system is closed under subject reduction can be illustrated by the following 'Cut and Paste' proof: Suppose that $B \vdash (\lambda x.M)N{:}\sigma$. By ($\rightarrow$E), there is a sequence $\sigma_1 \cap \cdots \cap \sigma_n$ such that

$$B \vdash \lambda x.M{:}\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma \text{ and } B \vdash N{:}\sigma_1 \cap \cdots \cap \sigma_n.$$

Since ($\rightarrow$I) should be the last step performed for the first result and ($\cap$I) should be the last step for the latter, also

$$B, x{:}\sigma_1 \cap \cdots \cap \sigma_n \vdash M{:}\sigma \text{ and } B \vdash N{:}\sigma_i, \text{ for } 1 \leq i \leq n.$$

Then a derivation for $B \vdash M[N/x]{:}\sigma$ can be obtained by replacing, for every $1 \leq i \leq n$, in the derivation for $B, x{:}\sigma_1 \cap \cdots \cap \sigma_n \vdash M{:}\sigma$, the sub-derivation $\{x{:}\sigma_1 \cap \cdots \cap \sigma_n\} \vdash x{:}\sigma_i$ by the derivation for $B \vdash N{:}\sigma_i$.

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

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

When restricting to λI-terms, the term-variable x occurs in M and the term N is a subterm of $M[N/x]$, so N is typed in the derivation for $B \vdash M[N/x]{:}\sigma$, probably with several different types $\sigma_1, \ldots, \sigma_n$. In the CD-system a derivation for

$$B, x{:}\sigma_1 \cap \cdots \cap \sigma_n \vdash M{:}\sigma$$

can be obtained by replacing in the derivation for $B \vdash M[N/x]{:}\sigma$, all occurrences of $N{:}\sigma_i$ by the derivation for $\{x{:}\sigma_1 \cap \cdots \cap \sigma_n\} \vdash x{:}\sigma_i$. Then, obviously, using (→I), (∩I), and (→E), the redex can be typed.

The system is not closed for β-expansion for the full λK-calculus: when the term-variable x does not occur in M, the term N is a not a subterm of $M[N/x]$, and if there is no ρ such that $B \vdash N{:}\rho$, the redex cannot be typed.

Take, for example, the two lambda terms $((\lambda xy.y)((\lambda z.zz)(\lambda z.zz)))$ and $(\lambda y.y)$ (notice that the first term reduces to the second). Then of course $\vdash \lambda y.y{:}\sigma \to \sigma$, but it is impossible to type the term $((\lambda xy.y.)((\lambda z.zz)(\lambda z.zz)))$. This can be understood from the fact that all typeable terms are normalizable; the subterm $((\lambda z.zz)(\lambda z.zz))$ has no normal form, so is not typeable.

For a more intricate example that remains within the set of typeable terms, take the two lambda terms $(\lambda yz.(\lambda b.z)(yz))$ and $(\lambda yz.z)$ (again the first term reduces to the second). It is easy to show that

$$\{z{:}\sigma, y{:}\tau\} \vdash z{:}\sigma,$$

but it is impossible to give a derivation for $(\lambda b.z)(yz){:}\sigma$ from the same basis. This is caused by the fact that $((\lambda b.z)(yz))$ can only be typed in this system from a basis in which the predicate for y is an arrow type scheme. It is for example possible to derive

$$\{z{:}\sigma, y{:}\sigma \to \tau\} \vdash (\lambda b.z)(yz){:}\sigma.$$

Therefore, it is possible to derive

$$\vdash \lambda yz.(\lambda b.z)(yz){:}(\sigma \to \tau) \to \sigma \to \sigma \text{ and } \vdash \lambda yz.z{:}\tau \to \sigma \to \sigma,$$

but not to give a derivation for $\lambda yz.(\lambda b.z)(yz){:}\tau \to \sigma \to \sigma$.

From this initial system several others emerged. The best known and most frequently quoted is the one presented in [6], but there are two earlier papers ([10] and [9]) that investigate interesting systems which can be regarded as in-between those of in [7] and [6]. In fact, the system that will be presented in Section *2* is more close to the system of [10] that to that of [6].

1.2 The Coppo-Dezani-Venneri type assignment systems

In [10] two type assignment systems are presented, that, in approach, are more general than the Coppo-Dezani system: in addition to the type constructors '→' and '∩', they also contain the type constant 'ω'. The first type discipline as presented in [10] (a similar system was presented in [35]) is a true extension of the one presented in [7]; the second one limits the use of intersection types in bases. By introducing the type constant ω next to the intersection types, a system is obtained that is closed under β-equality for the full λK-calculus: if $M =_\beta N$, where M and N are lambda terms, then $B \vdash M{:}\sigma \iff B \vdash N{:}\sigma$.

Definition 1.2.1 i) The set of types is inductively defined in [10] by:

a) All type-variables $\varphi_0, \varphi_1, \ldots$ are types.

b) ω is a type.

c) If $\sigma_1, \ldots, \sigma_n$ are types, $(n \geq 1)$, then $\sigma_1 \cap \cdots \cap \sigma_n$ is a sequence*.*

d) If $\sigma_1 \cap \cdots \cap \sigma_n$ is a sequence and σ is a type, then $\sigma_1 \cap \cdots \cap \sigma_n \to \sigma$ is a type.

ii) *Every type can be written as $\sigma_1{\rightarrow}\cdots{\rightarrow}\sigma_n{\rightarrow}\sigma$, where $\sigma_1, \ldots, \sigma_n$ are sequences, and σ is a type-variable or ω. The type is called tail-proper if $\sigma \neq \omega$.*

iii) *Type assignment and derivations are in [10] defined by:*

$$(\rightarrow I):\quad \frac{\begin{array}{c}[x{:}\sigma_i] \cdots [x{:}\sigma_n] \\ \vdots \\ M{:}\sigma\end{array}}{\lambda x.M{:}\tau{\rightarrow}\sigma}\ (a) \qquad (\cap I):\quad \frac{M{:}\sigma_1 \quad \cdots \quad M{:}\sigma_n}{M{:}\sigma_1{\cap}\cdots{\cap}\sigma_n}$$

$$(\omega):\quad \frac{}{M{:}\omega} \qquad (\rightarrow E):\quad \frac{M{:}\sigma_1{\cap}\cdots{\cap}\sigma_n{\rightarrow}\sigma \qquad N{:}\sigma_1{\cap}\cdots{\cap}\sigma_n}{MN{:}\sigma}$$

(a) If $x{:}\sigma_1, \ldots, x{:}\sigma_n$ are all and nothing but the statements about x on which $M{:}\sigma$ depends, and τ is a sequence that at least contains all $\sigma_1, \ldots, \sigma_n$.

Notice that ω is treated as a true type constant; the view of the strict intersection system presented in [1] (see Subsection *1.4*) and that of the essential system presented in this paper (see Section *2*) is to treat ω as the empty intersection.

The changes with respect to the system of [7] are small, but important. First of all, the type constant ω, assignable to all terms, is introduced, which makes all terms having a *head normal form* typeable; the type ω is needed to cover the subterms that have no normal form. Also, the ($\cap$E) rule of [7] is implicitly present in ($\rightarrow$I).

The main properties of this system proved in [10] are:

- If $B \vdash M{:}\sigma$ and $M =_\beta N$, then $B \vdash N{:}\sigma$.
- $B \vdash M{:}\sigma$ and σ is tail-proper, 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.

That the system is now also closed for β-expansion, is solved by the introduction of the type constant ω and the sequences. The type constant ω is the universal type, i.e. each term can be typed by ω. It can be used in the expansion from $B \vdash M[N/x]{:}\sigma$ to $B \vdash (\lambda x.M)N{:}\sigma$ to type N if N does not occur in $M[N/x]$, and there is no other type ρ such that $B \vdash N{:}\rho$. If N does not occur in $M[N/x]$, x does not occur in M, and there is a derivation for $B \vdash \lambda x.M{:}\omega{\rightarrow}\sigma$. Since N is typeable by ω, by derivation rule ($\rightarrow$E) also $B \vdash (\lambda x.M)N{:}\sigma$.

The sequences allow, as in the CD-system, for a term-variable to have different types within a derivation; they are used for the cases that N occurs more than once in $M[N/x]$, and these occurrences were typed in the derivation for $B \vdash M[N/x]{:}\sigma$ with different types. (See also Lemma *2.9*.)

Definition 1.2.2 *i) The set of normalized types is inductively defined in [10] by:*

a) All type-variables $\varphi_0, \varphi_1, \ldots$ are normalized types.

b) ω is a normalized sequence.

c) If $\sigma_1, \ldots, \sigma_n$ are normalized types $(n \geq 1)$ and, for $1 \leq i \leq n$, $\sigma_i \neq \omega$, then $\sigma_1{\cap}\cdots{\cap}\sigma_n$ is a normalized sequence.

d) If $\sigma_1{\cap}\cdots{\cap}\sigma_n$ is a normalized sequence and σ is a normalized type, where $\sigma \neq \omega$, then $\sigma_1{\cap}\cdots{\cap}\sigma_n{\rightarrow}\sigma$ is a normalized type.

ii) *On the set of types, the relation $\sim$ is defined by:*

a) $\varphi \sim \varphi$.

b) $\sigma_1 \cap \cdots \cap \sigma_i \cap \sigma_{i+1} \cap \cdots \cap \sigma_n \sim \sigma'_1 \cap \cdots \cap \sigma'_{i+1} \cap \sigma'_i \cap \cdots \cap \sigma'_n \iff$
$\forall\, 1 \leq i \leq n\, [\sigma_i \sim \sigma'_i]$.

c) $\sigma \to \tau \sim \sigma' \to \tau' \iff \sigma \sim \sigma' \;\&\; \tau \sim \tau'$.

Observe that the only normalized non-tail-proper type is ω.

A good justification for identification of types through $\sim$ can be found in the definition of type-semantics (Definition *1.3.5*). Notice that, in fact, normalizing types as done here is just the same as treating ω as an empty intersection; the set of normalized types coincides with the set of strict types (see Definition *1.4.1*), and normalized sequences correspond to strict intersection types.

The second type assignment system presented in [10] is a restricted version of the first. Since it limits the possible bases that can be used in a derivation, it is not a proper extension of Curry's system: if $B \vdash M{:}\sigma$ and the term-variable x does not occur in B, then for $(\lambda x.M)$ only the type $\omega \to \sigma$ can be derived. This system is also used in [19].

Definition 1.2.3 Restricted type assignment and restricted derivations are in [10] defined by:

$$(\to I):\quad \frac{\begin{array}{c}[x{:}\sigma_1] \cdots [x{:}\sigma_n] \\ \vdots \\ M{:}\sigma\end{array}}{\lambda x.M{:}\sigma_1 \cap \cdots \cap \sigma_n \to \sigma}\ (a) \qquad (\cap I):\quad \frac{M{:}\sigma_1 \quad \cdots \quad M{:}\sigma_n}{M{:}\sigma_1 \cap \cdots \cap \sigma_n}\ (b)$$

$$(\omega):\quad \frac{}{M{:}\omega} \qquad (\to E):\quad \frac{M{:}\sigma_1 \cap \cdots \cap \sigma_n \to \sigma \qquad N{:}\sigma_1 \cap \cdots \cap \sigma_n}{MN{:}\sigma}$$

(a) If $\sigma \neq \omega$ and $x{:}\sigma_1, \ldots, x{:}\sigma_n$ are all and nothing but the statements about x on which $M{:}\sigma$ depends. If n = 0, so in the derivation for $M{:}\sigma$ there is no premise whose subject is x, then $\sigma_1 \cap \cdots \cap \sigma_n = \omega$.

(b) If for $1 \leq i \leq n$, $\sigma_i \neq \omega$.

This notion of type assignment is *relevant* in the sense of [15]: in the ($\to$I)-rule, only those types actually used in the derivation can be abstracted. This implies that, for example, for the lambda term $(\lambda ab.a)$ the type $\sigma \to \tau \to \sigma$ cannot be derived.

It is obvious that $B \vdash M{:}\sigma$ in the restricted system implies $B \vdash M{:}\sigma$ in the unrestricted one and that the converse does not hold. In both systems, types are not invariant by η-expansion, since for example $\vdash \lambda x.x{:}\varphi \to \varphi$, but not $\vdash \lambda xy.xy{:}\varphi \to \varphi$. Moreover, type assignment in the unrestricted system is not invariant under η-reduction: for example

$$\vdash \lambda xy.xy{:}(\sigma \to \tau) \to \sigma \cap \rho \to \tau, \text{ but not } \vdash \lambda x.x{:}(\sigma \to \tau) \to \sigma \cap \rho \to \tau.$$

This is due to the fact that, in the unrestricted system, there is no way to obtain $x{:}\sigma \cap \rho \to \tau$ from $x{:}\sigma \to \tau$. In the restricted system, in the ($\to$I)-rule, only types actually used for a term variable can be collected. This means that the statement $\lambda xy.xy{:}(\sigma \to \tau) \to \sigma \cap \rho \to \tau$ cannot be derived, since the type ρ is not used for the application xy.

The closure under η-reduction, however, holds for the restricted system; properties of this restricted system proved in [10] are:

- If $B \vdash M{:}\sigma$, then σ is a normalized type.
- If $B \vdash M{:}\sigma$ and $M \to_\eta N$, then $B \vdash N{:}\sigma$.

1.3 The Barendregt-Coppo-Dezani type assignment system

The type assignment system presented in [6] by H. Barendregt, M. Coppo and M. Dezani-Ciancaglini is based on the first, unrestricted system as presented in [10]. It extended the set of types to $\mathcal{T}_\cap$, introduced a partial order relation '$\leq$' on types, added the type assignment rule ($\leq$), and introduced a more general form of the rules concerning intersection. The rule ($\leq$) was mainly introduced to prove completeness of type assignment.

In this paper, it was shown that the set of types derivable for a lambda term in the extended system is a filter, i.e. a set closed under intersection and right-closed for $\leq$. The interpretation of a lambda term by the set of types derivable for it – $[\![M]\!]_\xi$ – is defined in the standard way and gives a filter λ-model $\mathcal{F}_\cap$. The main result of that paper is that, using this model, completeness is proved by proving the statement: $\vdash_\cap M{:}\sigma \iff [\![M]\!] \in \upsilon(\sigma)$, where $\upsilon : \mathcal{T}_\cap \to \mathcal{F}_\cap$ is a simple type interpretation as defined in [23] (see Definition *1.3.5 (ii)*). In order to prove the '$\Leftarrow$'-part of this statement (completeness), the relation $\leq$ is needed. Other interesting applications of filter λ-models can be found in [8], [11], [16], and [17].

Definition 1.3.1 *i) $\mathcal{T}_\cap$, the set of types in [6] is inductively defined by:*

a) All type-variables $\varphi_0, \varphi_1, \ldots \in \mathcal{T}_\cap$.

b) $\omega \in \mathcal{T}_\cap$.

c) If σ and $\tau \in \mathcal{T}_\cap$, then $\sigma{\rightarrow}\tau$ and $\sigma{\cap}\tau \in \mathcal{T}_\cap$.

ii) On $\mathcal{T}_\cap$, the type inclusion relation $\leq$ is inductively defined by:

a) $\sigma \leq \sigma$.

b) $\sigma \leq \omega$.

c) $\omega \leq \omega{\rightarrow}\omega$.

d) $\sigma{\cap}\tau \leq \sigma$, $\sigma{\cap}\tau \leq \tau$.

e) $(\sigma{\rightarrow}\tau){\cap}(\sigma{\rightarrow}\rho) \leq \sigma{\rightarrow}\tau{\cap}\rho$.

f) $\sigma \leq \tau \leq \rho \Rightarrow \sigma \leq \rho$.

g) $\sigma \leq \tau$ & $\sigma \leq \rho \Rightarrow \sigma \leq \tau{\cap}\rho$.

h) $\rho \leq \sigma$ & $\tau \leq \mu \Rightarrow \sigma{\rightarrow}\tau \leq \rho{\rightarrow}\mu$.

iii) $\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$.

$\mathcal{T}_\cap$ may be considered modulo $\sim$. Then $\leq$ becomes a partial order.

Notice that, because of part *(i.c)*, '$\cap$' is now also a general type constructor. This is in harmony with the fact that also ω is treated as a 'normal' type constant, but is not motivated by the type system itself. The decision to treat '$\cap$' in this way stems from a, in the eyes of this author, wrong line of thought. In order to solve the problem of completeness of type assignment, first the set of types is expanded, where the introduction of a relation on normalized types with contra-variance in the arrow would have done the trick. The goal of this paper is to show that, had the authors restricted themselves to the normalized types of [10] (Definition *1.2.2*) and introduced a relation $\leq$ on those types (essentially composed of parts *(ii.a)*, *(ii.b)*, *(ii.d)*, *(ii.f)*, *(ii.g)*, and *(ii.h)*), they would have obtained a system with the same expressiveness as that one they defined. This system would then have been identical to the essential system as presented in this paper (Section *2*).

Definition 1.3.2 *i) Type assignment and derivations are in [6] defined by the following natural deduction system.*

$$(\to E):\ \frac{M{:}\sigma\to\tau \qquad N{:}\sigma}{MN{:}\tau} \qquad\qquad (\to I):\ \frac{\begin{matrix}[x{:}\sigma]\\ \vdots \\ M{:}\tau\end{matrix}}{\lambda x.M{:}\sigma\to\tau}\ (a)$$

$$(\cap E):\ \frac{M{:}\sigma\cap\tau}{M{:}\sigma} \quad \frac{M{:}\sigma\cap\tau}{M{:}\tau} \qquad\qquad (\cap I):\ \frac{M{:}\sigma \qquad M{:}\tau}{M{:}\sigma\cap\tau}$$

$$(\omega):\ \frac{}{M{:}\omega} \qquad\qquad (\leq):\ \frac{M{:}\sigma \qquad \sigma\leq\tau}{M{:}\tau}$$

(a) If $x{:}\sigma$ is the only statement about x on which $M{:}\tau$ depends.

ii) $B \vdash_\cap M{:}\sigma$ means that $M{:}\sigma$ is derivable from a basis B using these rules.

An advantage of the presentation in [6] is, clearly, a very easy type definition and easy understandable derivation rules. But this advantage is superficial, since all difficulties now show up while proving theorems; especially the complexity of $\leq$ and $\sim$ causes confusion.

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

Lemma 1.3.3 *i)* $B \vdash_\cap MN{:}\sigma \iff \exists\,\tau\ [B \vdash_\cap M{:}\tau\to\sigma\ \&\ B \vdash_\cap N{:}\tau]$.

ii) $B \vdash_\cap \lambda x.M{:}\sigma \iff \exists\,\rho,\mu\ [\sigma = \rho\to\mu\ \&\ B, x{:}\rho \vdash_\cap M{:}\mu]$.

iii) $B \vdash_\cap x{:}\sigma \iff \exists\,\rho\ [x{:}\rho\in B\ \&\ \rho\leq\sigma]$. ■

In the BCD-system, there are several ways to deduce a desired result, due to the presence of the derivation rules (∩I), (∩E) and ($\leq$), which allow superfluous steps in derivations. In the CDV-systems, these rules are not present and there is a one-one relationship between terms and skeletons of derivations. In other words: those systems are syntax directed. The BCD-system has the same expressive power as the previous unrestricted CDV-system: all solvable terms have types other than ω, and a term has a normal form if and only if it has a type without ω-occurrences.

The main result of [6] is the proof for completeness of type assignment. The construction of a filter λ-model and the definition of a map from types to elements of this model (a *simple type interpretation*) make that proof possible: if the interpretation of the term M is an element of the interpretation of the type σ, then M is typeable with σ.

Filters and the filter λ-model $\mathcal{F}_\cap$ are defined by:

Definition 1.3.4 *i) A filter is a subset $d \subseteq \mathcal{T}_\cap$ such that:*

a) $\omega\in d$.

b) $\sigma,\tau\in d \Rightarrow \sigma\cap\tau\in d$.

c) $\sigma\geq\tau\in d \Rightarrow \sigma\in d$.

ii) $\mathcal{F}_\cap = \{d \mid d \text{ is a filter}\}$.

iii) For $d_1, d_2\in\mathcal{F}_\cap$ define $d_1\cdot d_2 = \{\tau\in\mathcal{T}_\cap \mid \exists\,\sigma\in d_2\ [\sigma\to\tau\in d_1]\}$.

Notice that, because of part *(i.a)*, a filter is never empty.

The following properties are proved in [6]:

- $\forall M \in \Lambda\, [\{\sigma \mid \exists B\, [B \vdash_\cap M{:}\sigma\} \in \mathcal{F}_\cap]]$.
- Let ξ be a valuation of term-variables in $\mathcal{F}_\cap$ and $B_\xi = \{x{:}\sigma \mid \sigma \in \xi\,(x)\}$. For $M \in \Lambda$ define $[\![M]\!]_\xi = \{\sigma \mid B_\xi \vdash_\cap M{:}\sigma\}$. Using the definition of Hindley and Longo [24] it is shown that $\langle \mathcal{F}_\cap, \cdot, [\![\ ]\!]\rangle$ is a λ-model.

In constructing a complete system, the semantics of types plays a crucial role. As in [17], [29], and essentially following [22], a distinction can be made between several notions of type interpretations and semantic satisfiability. There are, roughly, three notions of type semantics that differ in the meaning of an arrow type scheme: inference type interpretations, simple type interpretations and F type interpretations. These different notions of type interpretations induce of course different notions of semantic satisfiability.

Definition 1.3.5 *i) Let $\langle \mathcal{D}, \cdot, \varepsilon\rangle$ be a continuous λ-model. A mapping $\upsilon : \mathcal{T}_\cap \to \wp(\mathcal{D}) = \{X \mid X \subseteq \mathcal{D}\}$ is a type interpretation if and only if:*
a) $\{\varepsilon \cdot d \mid \forall e \in \upsilon(\sigma)\ [d \cdot e \in \upsilon(\tau)]\} \subseteq \upsilon(\sigma \to \tau)$.
b) $\upsilon(\sigma \to \tau) \subseteq \{d \mid \forall e \in \upsilon(\sigma)\ [d \cdot e \in \upsilon(\tau)]\}$.
c) $\upsilon(\sigma \cap \tau) = \upsilon(\sigma) \cap \upsilon(\tau)$.
ii) Following [23], a type interpretation is simple if also:

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

iii) A type interpretation is called an F type interpretation if it satisfies:

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

Notice that, in part *(ii)*, the containment relation $\subseteq$ of part *(i.b)* is replaced by =, and that in part *(iii)* the same is done with regard to part *(i.a)*.

These notions of type interpretation lead, naturally, to the following definitions for semantic satisfiability (called *inference-*, *simple-* and *F-semantics*, respectively).

Definition 1.3.6 *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) 1) $B \models M{:}\sigma \iff \forall \mathcal{M}, \xi, \upsilon\ [\mathcal{M}, \xi, \upsilon \models B \Rightarrow \mathcal{M}, \xi, \upsilon \models M{:}\sigma]$.
2) $B \models_s M{:}\sigma \iff \forall \mathcal{M}, \xi$, simple type interpretations υ

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

3) $B \models_F M{:}\sigma \iff \forall \mathcal{M}, \xi$, F type interpretations υ

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

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

The main result of [6] is obtained by proving:

Property 1.3.7 *i)* Soundness. $B \vdash_\cap M{:}\sigma \Rightarrow B \models_s M{:}\sigma$.
ii) Completeness. $B \models_s M{:}\sigma \Rightarrow B \vdash_\cap M{:}\sigma$. ■

The proof of completeness is obtained in a way very similar to that of Theorem *4.3.6*. The results of [6] in fact show that type assignment in the BCD-system is complete with respect to simple type semantics; this in contrast to strict type assignment (presented in [1], see also the next subsection), that is complete with respect to the inference semantics.

1.4 The strict type assignment system

The strict type assignment system as defined in [1] is a restriction of the system of [6]; it is a type assignment system in which the relation $\leq$ and the derivation rule $(\leq)$ are no longer present. The elimination of $\leq$ induces a set of strict types, that is actually the set of normalized tail-proper types of [10]. Moreover, there the relation $\leq_{\mathrm{S}}$ on strict types is presented, that is more restricted than just the relation $\leq$ restricted to strict types. Instead, it is the relation generated by interpreting the type-constructor $\cap$ as intersection on sets; in particular, $\leq_{\mathrm{S}}$ is not defined over '$\rightarrow$'-types, so part *(ii.h)* of Definition *1.3.1* is missing. The derivation rules used are similar to those of the unrestricted system in [10]. This implies that, formally, the strict system is a system in between the two presented in [10]; a significant difference with the restricted system of [10] is that type assignment in the strict system is *not* relevant in the sense of [15].

Although the rather strong restrictions imposed, the provable results for the strict system are very close to those for the system of [6]. 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 $\mathcal{F}_{\mathrm{S}}$, that satisfies all major properties of the filter λ-model $\mathcal{F}_{\cap}$ as presented in [6], but is an essentially different λ-model, equivalent to Engeler's model $\mathcal{D}_A$ [18]. In [1] was shown that soundness for the notion of type assignment of [6] is lost if instead of simple type semantics, the inference type semantics is used. Take, for example, the statement $\lambda x.x{:}(\sigma\rightarrow\sigma)\rightarrow(\sigma\cap\tau)\rightarrow\sigma$: this statement is derivable in the system $\vdash_{\cap}$, but it is not valid in $\mathcal{F}_{\mathrm{S}}$. 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 $\leq$; this was done using $\mathcal{F}_{\mathrm{S}}$.

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.

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. Perhaps the most important change from the systems presented before is that the type constant ω is no longer treated as an arbitrary type, that can be handled in any way. Instead, ω is taken to be the empty intersection: if $n = 0$, then $\sigma_1\cap\cdots\cap\sigma_n \equiv \omega$, 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 [7], [9], and [10].

Definition 1.4.1 *i)* $\mathcal{T}_{\mathrm{s}}$, *the set of* strict types, *and* $\mathcal{T}_{\mathrm{S}}$, *the set of* strict intersection types, *are defined by mutual induction by:*

a) All type-variables $\varphi_0, \varphi_1, \ldots \in \mathcal{T}_{\mathrm{s}}$, *and if* $\tau \in \mathcal{T}_{\mathrm{s}}$ *and* $\sigma \in \mathcal{T}_{\mathrm{S}}$, *then* $\sigma\rightarrow\tau \in \mathcal{T}_{\mathrm{s}}$.

b) If $\sigma_1, \ldots, \sigma_n \in \mathcal{T}_{\mathrm{s}}$ $(n \geq 0)$, *then* $\sigma_1\cap\cdots\cap\sigma_n \in \mathcal{T}_{\mathrm{S}}$.

ii) On $\mathcal{T}_{\mathrm{S}}$, *the relation* $\leq_{\mathrm{S}}$ *is defined by:*

a) $\forall\, 1 \leq i \leq n\ (n \geq 1)\ [\sigma_1\cap\cdots\cap\sigma_n \leq_{\mathrm{S}} \sigma_i]$.

b) $\forall\, 1 \leq i \leq n\ (n \geq 0)\ [\sigma \leq_{\mathrm{S}} \sigma_i] \Rightarrow \sigma \leq_{\mathrm{S}} \sigma_1\cap\cdots\cap\sigma_n$.

c) $\sigma \leq_{\mathrm{S}} \tau \leq_{\mathrm{S}} \rho \Rightarrow \sigma \leq_{\mathrm{S}} \rho$.

iii) On $\mathcal{T}_{\mathrm{S}}$, *the relation* $\sim_{\mathrm{S}}$ *is defined by:*

a) $\sigma \leq_S \tau \leq_S \sigma \Rightarrow \sigma \sim_S \tau$.

b) $\sigma \sim_S \rho \;\&\; \tau \sim_S \mu \Rightarrow \sigma \rightarrow \tau \sim_S \rho \rightarrow \mu$.

Notice that $\mathcal{T}_s$ is a proper subset of $\mathcal{T}_S$, and that the second part of *(i.a)* can also be formulated as: If $\sigma, \sigma_1, \ldots, \sigma_n \in \mathcal{T}_s$ $(n \geq 0)$, then $\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma \in \mathcal{T}_s$. Moreover, $\sigma \sim_S \rho$ if and only if σ can be obtained from ρ by premuting components of an intersection subtype, e.g. in $\sigma \cap \rho \rightarrow \tau \sim_S \rho \cap \sigma \rightarrow \tau$.

The definition of $\sim_S$ as in [1] did not contain part *1.4.1 (iii.b)*, but was defined by: $\sigma \leq_S \tau \leq_S \sigma \Leftrightarrow \sigma \sim_S \tau$. As was remarked by Professor G. Plotkin of the University of Edinburgh, Scotland (private communication), defining the equivalence relation on types in that way causes an anomaly in the definition of type-interpretation. In particular, the types $\sigma \cap \rho \rightarrow \tau$ and $\rho \cap \sigma \rightarrow \tau$ would be incomparable, which implies that the interpretation of an arrow type $\sigma \rightarrow \tau$ is no longer a map from the interpretation of σ onto the interpretation of τ.

Definition 1.4.2 *i) Strict type assignment and strict derivations are defined by the following natural deduction system (where all types displayed are strict, except for σ in the rule $(\rightarrow I)$):*

$$(\rightarrow E): \quad \frac{M{:}\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma \qquad N{:}\sigma_1 \ldots N{:}\sigma_n}{MN{:}\sigma} \quad (n \geq 0)$$

$$(\rightarrow I): \quad \frac{\begin{array}{c}[x{:}\sigma] \\ \vdots \\ M{:}\tau\end{array}}{\lambda x.M{:}\sigma \rightarrow \tau} \;\; (a) \qquad\qquad (\cap E): \quad \frac{x{:}\sigma_1 \cap \cdots \cap \sigma_n}{x{:}\sigma_i} \;\; (b)$$

(a) If $x{:}\sigma$ is the only statement about x on which $M{:}\tau$ depends.

(b) $n \geq 2, 1 \leq i \leq n$.

ii) $B \vdash_s M{:}\sigma$ is used for: $M{:}\sigma$ is derivable from B using a strict derivation, and $\vdash_S$ is defined by: $B \vdash_S M{:}\sigma$ if and only if: there are $\sigma_1, \ldots, \sigma_n$ $(n \geq 0)$ such that $\sigma \equiv \sigma_1 \cap \cdots \cap \sigma_n$, and for every $1 \leq i \leq n$, $B \vdash_s M{:}\sigma_i$.

Notice that the derivation rule $(\cap E)$ is only performed on variables and that the derivation rules (ω) and $(\cap I)$ of the BCD-system are implicitly present in the derivation rule $(\rightarrow E)$. Moreover, a derivation in the $\vdash_S$ system with conclusion $M{:}\omega$ cannot be composed with any other derivation. The derivation rule $(\cap E)$ could be replaced by a rule for $(\leq_S)$, in the spirit of the rule $(\leq)$ of the BCD-system, but allowed only for term-variables. Also, the derivation rules $(\rightarrow I)$ and $(\cap E)$ together correspond to the derivation rule $(\rightarrow I)$ of the unrestricted CDV-system (Definition *1.2.1*).

The strict filter λ-model $\mathcal{F}_S$ is defined in a way very similar to $\mathcal{F}_\cap$ defined in [6], by defining filters of types and a map from terms to filters.

Definition 1.4.3 *i) A subset d of $\mathcal{T}_S$ is called a strict filter if and only if:*

a) $\sigma_1, \ldots, \sigma_n \in d$ $(n \geq 0) \Rightarrow \sigma_1 \cap \cdots \cap \sigma_n \in d$.

b) $\tau \in d \;\&\; \tau \leq_S \sigma \Rightarrow \sigma \in d$.

ii) If $V \subseteq \mathcal{T}_S$, then $\uparrow_S V$ is the smallest strict filter that contains V, and $\uparrow_S \sigma = \uparrow_S \{\sigma\}$.

iii) $\mathcal{F}_S = \{d \subseteq \mathcal{T}_S \mid d$ is a strict filter$\}$. Application on $\mathcal{F}_S$ is defined by:

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

Notice that if types are not considered modulo $\sim_{\rm S}$ then part *(i.b)* should also contain: $\tau \in d$ & $\tau \sim_{\rm S} \sigma \Rightarrow \sigma \in d$. Notice also that every strict filter contains ω.

The filter λ-models $\mathcal{F}_{\rm S}$ and $\mathcal{F}_\cap$ are not isomorphic as complete lattices, since, for example, in $\mathcal{F}_\cap$ the filter $\uparrow(\sigma\cap\tau)\rightarrow\sigma$ is contained in $\uparrow\sigma\rightarrow\sigma$, but in $\mathcal{F}_{\rm S}$ the strict filter $\uparrow_{\rm S}(\sigma\cap\tau)\rightarrow\sigma$ is not contained in $\uparrow_{\rm S}\sigma\rightarrow\sigma$. Moreover, they are not isomorphic as λ-models since in $\mathcal{F}_\cap$ the meaning of $(\lambda xy.xy)$ is contained in the meaning of $(\lambda x.x)$, while this does not hold in $\mathcal{F}_{\rm S}$. Another difference is that, while the analogue of G in $\mathcal{F}_\cap$ chooses the minimal representative of functions, this is not the case in $\mathcal{F}_{\rm S}$. Moreover, it is straightforward to show that $\mathcal{F}_{\rm S}$ is equivalent to Engeler's model $\mathcal{D}_A$.

The main results of [1] are:

- Soundness and completeness of type assignment with respect to *inference* type semantics.
- If $B \vdash_\cap M{:}\sigma$ then there are $B', \sigma' \in \mathcal{T}_{\rm S}$ such that $B' \vdash_{\rm S} M{:}\sigma'$, $\sigma' \leq \sigma$ and $B \leq B'$.
- *Conservativity.* Let $B, \sigma \in \mathcal{T}_{\rm S}$. If $B \vdash_\cap M{:}\sigma$, then $B \vdash_{\rm S} M{:}\sigma$.
- In the BCD-system without ω, restricted to the λI-calculus: $B \models_{\rm s} M{:}\sigma \iff B \vdash_\cap M{:}\sigma$.
- In the BCD-system without ω: $\{M \mid M$ is typeable by means of the derivation rules ($\cap$I), ($\cap$E), ($\rightarrow$I) and ($\rightarrow$E)$\}$ = $\{M \mid M$ is strongly normalizable$\}$.

2 The essential intersection type assignment system

In this section the essential type assignment system is presented, a restricted version of the system presented in [6], 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] (see Subsection *1.4*); the main difference is that the strict system is not closed for η-reduction, whereas the essential system presented here is.

Recall Definition *1.4.1*. The relation $\leq_{\rm E}$ on $\mathcal{T}_{\rm S}$, to be defined below, is a natural extension of the relation $\leq_{\rm S}$, that was only defined for intersection types. Notice that, in the definition of $\leq_{\rm E}$, the arrow type constructor is contra-variant in its left-hand argument.

Definition 2.1 *i) The relation $\leq_{\rm E}$ is defined on $\mathcal{T}_{\rm S}$ like $\leq_{\rm S}$, by adding the last alternative.*

a) $\forall 1 \leq i \leq n \ (n \geq 1)\ [\sigma_1\cap\cdots\cap\sigma_n \leq_{\rm E} \sigma_i]$.

b) $\forall 1 \leq i \leq n \ (n \geq 0)\ [\sigma \leq_{\rm E} \sigma_i] \Rightarrow \sigma \leq_{\rm E} \sigma_1\cap\cdots\cap\sigma_n$.

c) $\sigma \leq_{\rm E} \tau \leq_{\rm E} \rho \Rightarrow \sigma \leq_{\rm E} \rho$.

d) $\rho \leq_{\rm E} \sigma$ *&* $\tau \leq_{\rm E} \mu \Rightarrow \sigma\rightarrow\tau \leq_{\rm E} \rho\rightarrow\mu$.

ii) On $\mathcal{T}_{\rm S}$, the relation $\sim_{\rm E}$ is defined by: $\sigma \sim_{\rm E} \tau \iff \sigma \leq_{\rm E} \tau \leq_{\rm E} \sigma$.

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

Also the relations $\leq$, $\sim$, and $\leq_{\rm S}$ are extended to bases.

Notice that $\leq_{\rm E}$ is exactly the relation $\leq$ restricted to $\mathcal{T}_{\rm S}$ (see also Property *4.1.2*), so if $\sigma \leq_{\rm E} \tau$, then $\sigma \leq \tau$, and that $\sim_{\rm S}$ is a true subrelation of $\sim_{\rm E}$, since, for example, $(\sigma\rightarrow\tau)\cap(\sigma\cap\rho\rightarrow\tau) \sim_{\rm E} \sigma\rightarrow\tau$, but this does not hold in $\sim_{\rm S}$. Moreover, $\mathcal{T}_{\rm S}$ may be considered modulo $\sim_{\rm E}$; then $\leq_{\rm E}$ becomes a partial order, and from now on in this paper types are considered modulo $\sim_{\rm E}$.

Unless stated otherwise, if a type is written as $\sigma_1\cap\cdots\cap\sigma_n$, then all $\sigma_1, \ldots, \sigma_n$ are assumed to be strict. Remember that ω is defined to be the emtpy intersection.

Definition 2.2 If $B_1, \ldots, B_n$ are bases, then $\Pi\{B_1, \ldots, B_n\}$ is the basis defined as follows: $x{:}\sigma_1\cap\cdots\cap\sigma_m \in \Pi\{B_1, \ldots, B_n\}$ if and only if $\{x{:}\sigma_1, \ldots, x{:}\sigma_m\}$ is the set of all statements about x that occur in $B_1 \cup \ldots \cup B_n$.

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

For the relation $\leq_{\rm E}$, the following properties hold:

Lemma 2.3 *i)* $\sigma \leq_{\rm S} \tau \Rightarrow \sigma \leq_{\rm E} \tau$.

ii) $\varphi \leq_{\rm E} \sigma \iff \sigma \equiv \varphi$. *So* $\{\sigma \mid \sigma \sim_{\rm E} \varphi\} = \{\varphi\}$.

iii) $\omega \leq_{\rm E} \sigma \iff \sigma \equiv \omega$. *So* $\{\sigma \mid \sigma \sim_{\rm E} \omega\} = \{\omega\}$.

iv) $\sigma \to \tau \leq_{\rm E} \rho \in \mathcal{T}_{\rm s} \iff \exists\, \alpha \in \mathcal{T}_{\rm S}, \beta \in \mathcal{T}_{\rm s}\, [\rho \equiv \alpha \to \beta \;\&\; \alpha \leq_{\rm E} \sigma \;\&\; \tau \leq_{\rm E} \beta]$.

v) $\sigma_1\cap\cdots\cap\sigma_n \leq_{\rm E} \tau \in \mathcal{T}_{\rm s} \Rightarrow \exists\, 1 \leq i \leq n\, [\sigma_i \leq_{\rm E} \tau]$.

vi) $\sigma \leq_{\rm E} \tau \Rightarrow \exists\, \sigma_1, \ldots, \sigma_n, \tau_1, \ldots, \tau_m\, [\sigma = \sigma_1\cap\cdots\cap\sigma_n \;\&\; \tau = \tau_1\cap\cdots\cap\tau_m \;\&\; \forall\, 1 \leq j \leq m\; \exists\, 1 \leq i \leq n\, [\sigma_i \leq_{\rm E} \tau_j]]$.

Proof: Easy. ■

The essential type assignment system is constructed from the set of strict types and an extension of the derivation rules as in Definition *1.4.2*. In this way, a syntax directed system is obtained, that satisfies the main properties of the BCD-system.

Definition 2.4 *i) Essential type assignment and essential derivations are defined by the following natural deduction system (where all types displayed are strict, except σ in the rules ($\to$I) and ($\leq_{\rm E}$)):*

$$(\to E):\quad \frac{M{:}\sigma_1\cap\cdots\cap\sigma_n\to\tau \qquad N{:}\sigma_1 \ldots\; N{:}\sigma_n}{MN{:}\tau}\quad (n \geq 0)$$

$$(\to I):\quad \frac{\begin{array}{c}[x{:}\sigma]\\ \vdots \\ M{:}\tau\end{array}}{\lambda x.M{:}\sigma\to\tau}\ (a) \qquad\qquad (\leq_{\rm E}):\quad \frac{x{:}\sigma \qquad \sigma \leq_{\rm E} \tau}{x{:}\tau}$$

(a) If $x{:}\sigma$ is the only statement about x on which $M{:}\tau$ depends.

$B \vdash_{\rm e} M{:}\sigma$ is defined as: $M{:}\sigma$ is derivable from B using an essential derivation.

ii) Define $\vdash_{\rm E}$ by: $B \vdash_{\rm E} M{:}\sigma$ if and only if: there are $\sigma_1, \ldots, \sigma_n$ $(n \geq 0)$ such that $\sigma \equiv \sigma_1\cap\cdots\cap\sigma_n$ and $B \vdash_{\rm e} M{:}\sigma_i$, for every $1 \leq i \leq n$.

Notice that the difference between the strict system and the essential one lies only in the derivation rule for term variables. Instead of a rule ($\cap$E), that is in fact defined using the relation $\leq_{\rm S}$, the essential system contains a similar rule using $\leq_{\rm E}$.

The introduction of two different notions of derivability seems somewhat superfluous. In fact, the 'real' notion of type assignment is that defined as $\vdash_{\rm e}$; the symbol $\vdash_{\rm E}$ is mainly introduced for convenience, as an abbreviation.

For these notions of type assignment, the following properties hold:

Lemma 2.5 *i)* $B \vdash_{\rm E} x{:}\sigma \iff \exists\, \rho \in \mathcal{T}_{\rm S}\, [x{:}\rho \in B \;\&\; \rho \leq_{\rm E} \sigma]$.

ii) $B \vdash_{\text{e}} MN{:}\sigma \iff \exists \tau \in \mathcal{T}_{\text{S}} \, [B \vdash_{\text{e}} M{:}\tau \to \sigma \;\&\; B \vdash_{\text{E}} N{:}\tau]$.

iii) $B \vdash_{\text{e}} \lambda x.M{:}\sigma \iff \exists \rho \in \mathcal{T}_{\text{S}}, \mu \in \mathcal{T}_{\text{s}} \, [\sigma = \rho \to \mu \;\&\; B, x{:}\rho \vdash_{\text{e}} M{:}\mu]$.

iv) $B \vdash_{\text{e}} M{:}\sigma \iff B \vdash_{\text{E}} M{:}\sigma \;\&\; \sigma \in \mathcal{T}_{\text{s}}$.

v) $B \vdash_{\text{E}} M{:}\sigma \iff \exists \sigma_1, \ldots, \sigma_n \, [\sigma = \sigma_1 \cap \cdots \cap \sigma_n \;\&\; \forall 1 \leq i \leq n \, [B \vdash_{\text{e}} M{:}\sigma_i]]$.

vi) $B \vdash_{\text{E}} M{:}\sigma \iff \{x{:}\tau \in B \,|\, x \in FV(M)\} \vdash_{\text{E}} M{:}\sigma$.

vii) $B \vdash_{\text{E}} M{:}\sigma \;\&\; B' \leq_{\text{E}} B \Rightarrow B' \vdash_{\text{E}} M{:}\sigma$.

viii) $B \vdash_{\text{E}} M{:}\sigma \;\&\; M \sqsubseteq\!\!\!\!\!\sim M' \Rightarrow B' \vdash_{\text{E}} M'{:}\sigma$.

Proof: Easy. ■

Although the rule ($\leq_{\text{E}}$) is defined only for term-variables, $\vdash_{\text{E}}$ is closed for $\leq_{\text{E}}$.

Lemma 2.6 If $B \vdash_{\text{E}} M{:}\sigma$ and $\sigma \leq_{\text{E}} \tau$, then $B \vdash_{\text{E}} M{:}\tau$, so the following is an admissible rule in $\vdash_{\text{E}}$:

$$(\leq_{\text{E}}): \quad \frac{M{:}\sigma \qquad \sigma \leq_{\text{E}} \tau}{M{:}\tau}$$

Proof: By induction on $\vdash_{\text{E}}$.

i) $\sigma = \omega$. Then, by Lemma *2.3 (iii)*, $\tau = \omega$. Obviously, $B \vdash_{\text{E}} M{:}\tau$.

ii) $\sigma = \sigma_1 \cap \cdots \cap \sigma_n$. By Lemma *2.5 (v)*, for every $1 \leq i \leq n$, $B \vdash_{\text{e}} M{:}\sigma_i$. By Lemma *2.3 (vi)*, there are $\tau_1, \ldots, \tau_m \in \mathcal{T}_{\text{s}}$ such that $\tau = \tau_1 \cap \cdots \cap \tau_m$ and, for every $1 \leq j \leq m$, there is a $1 \leq i \leq n$ such that $\sigma_i \leq_{\text{E}} \tau_j$. By induction, for every $1 \leq j \leq m$, $B \vdash_{\text{e}} M{:}\tau_j$. But then, by Lemma *2.5 (v)*, $B \vdash_{\text{E}} M{:}\tau$.

iii) $\sigma \in \mathcal{T}_{\text{s}}$. This part is proven by induction on M.

a) $M \equiv x$. Then $B \leq_{\text{E}} \{x{:}\sigma\} \leq_{\text{E}} \{x{:}\tau\}$, so, by Lemma *2.5 (i)*, $B \vdash_{\text{E}} x{:}\tau$.

b) $M \equiv \lambda x.M'$. Then, by Lemma *2.5 (iii)*, there are $\rho \in \mathcal{T}_{\text{S}}$, $\mu \in \mathcal{T}_{\text{s}}$ such that $\sigma = \rho \to \mu$ and $B, x{:}\rho \vdash_{\text{e}} M'{:}\mu$. By Lemmas *2.3 (vi)* and *2.3 (iv)* there are $\rho_1, \ldots, \rho_n, \mu_1, \ldots, \mu_n$ such that $\tau = (\rho_1 \to \mu_1) \cap \cdots \cap (\rho_n \to \mu_n)$, and for $1 \leq i \leq n$, $\rho_i \leq_{\text{E}} \rho$ and $\mu \leq_{\text{E}} \mu_i$. By Lemma *2.5 (vii)*, for $1 \leq i \leq n$, $B, x{:}\rho_i \vdash_{\text{e}} M'{:}\mu$, and by induction $B, x{:}\rho_i \vdash_{\text{e}} M'{:}\mu_i$. So, by Lemma *2.5 (iii)*, for every $1 \leq i \leq n$, $B \vdash_{\text{e}} \lambda x.M'{:}\rho_i \to \mu_i$, so, to conclude, by Lemma *2.5 (v)*, $B \vdash_{\text{E}} \lambda x.M'{:}\tau$.

c) $M \equiv M_1 M_2$. Then, by Lemma *2.5 (ii)*, there is a $\mu \in \mathcal{T}_{\text{S}}$ such that $B \vdash_{\text{e}} M_1{:}\mu \to \sigma$ and $B \vdash_{\text{E}} M_2{:}\mu$. Since $\sigma \leq_{\text{E}} \tau$, also $\mu \to \sigma \leq_{\text{E}} \mu \to \tau$ and, by induction, $B \vdash_{\text{e}} M_1{:}\mu \to \tau$. Then, by Lemma *2.5 (ii)*, $B \vdash_{\text{E}} M_1 M_2{:}\tau$. ■

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 *2.7* and Theorem *2.8*. The lemma is also used in the proof of Lemmas *3.2.1*, Theorem *3.2.5*, and Theorem *6.2.15*.

Lemma 2.7 $B, x{:}\sigma \vdash_{\text{e}} Mx{:}\tau \;\&\; x \notin FV(M) \Rightarrow B \vdash_{\text{e}} M{:}\sigma \to \tau$.

Proof: $B, x{:}\sigma \vdash_{\text{e}} Mx{:}\tau \;\&\; x \notin \text{FV}(M) \Rightarrow$ (*2.5 (ii)*)

$\exists \mu \, [B, x{:}\sigma \vdash_{\text{e}} M{:}\mu \to \tau \;\&\; B, x{:}\sigma \vdash_{\text{E}} x{:}\mu] \Rightarrow$ (*2.5 (i)*, x not in M)

$\exists \mu \, [B \vdash_{\text{e}} M{:}\mu \to \tau \;\&\; \sigma \leq_{\text{E}} \mu] \Rightarrow$ (*2.1*)

$\exists \mu \, [B \vdash_{\text{e}} M{:}\mu \to \tau \;\&\; \mu \to \tau \leq_{\text{E}} \sigma \to \tau] \Rightarrow$ (*2.6*)

$B \vdash_{\text{e}} M{:}\sigma \to \tau$. ■

Theorem 2.8 If $B \vdash_{\text{E}} M{:}\sigma$ and $M \to_\eta N$, then $B \vdash_{\text{E}} N{:}\sigma$.

Proof: Only the part $\sigma \in \mathcal{T}_{\text{s}}$ is shown. The proof is completed 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. Then: $B \vdash_{\mathrm{e}} \lambda x.Mx{:}\sigma \;\Rightarrow\; (2.5\,(iii))$
$\exists\, \rho, \mu\; [\sigma = \rho{\to}\mu \;\&\; B, x{:}\rho \vdash_{\mathrm{e}} Mx{:}\mu] \;\Rightarrow\; (2.7) \quad B \vdash_{\mathrm{e}} M{:}\sigma.$ ■

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

As in [9, 6, 1], it is possible to prove that the essential type assignment system is closed under $=_\beta$. In the first paper, this result was obtained by a 'Cut and Paste'-proof (see Subsection *1.1*). In the latter two papers this result was obtained by building a filter λ-model as sketched in Section *1.3*; 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 *4.3.5*). 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 '∩' and the type constant ω can be made apparent.

First, a substitution lemma is proved. Notice that, unlike for many other notions of type assignment (Curry's system, the CD-system, the polymorphic type discipline [20]), the implication holds in both directions.

Lemma 2.9 $\exists\, \rho\; [B, x{:}\rho \vdash_{\mathrm{E}} M{:}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho] \iff B \vdash_{\mathrm{E}} M[N/x]{:}\sigma.$

Proof: By induction on M. Only the case $\sigma \in \mathcal{T}_{\mathrm{s}}$ is considered, of which only the non-trivial parts are shown.

i) $M \equiv x$.
⇒) $\exists\, \rho\; [B, x{:}\rho \vdash_{\mathrm{e}} x{:}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho] \;\Rightarrow$ (*2.5 (i)*)
$\exists\, \rho\; [\rho \leq_{\mathrm{E}} \sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho] \;\Rightarrow$ (*2.5 (iv)* & *2.6*)
$B \vdash_{\mathrm{e}} x[N/x]{:}\sigma.$

ii) $M \equiv y \neq x$.
⇐) $B \vdash_{\mathrm{e}} y[N/x]{:}\sigma \;\Rightarrow\; B \vdash_{\mathrm{e}} y{:}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\omega.$

iii) $M \equiv \lambda y.M'$. By Lemma *2.5 (iii)* and induction.

iv) $M \equiv M_1M_2$.
⇐) $B \vdash_{\mathrm{e}} M_1M_2[N/x]{:}\sigma \;\Rightarrow$ (*2.5 (ii)* & IH)
$\exists\, \rho_1, \rho_2, \tau\; [B, x{:}\rho_i \vdash_{\mathrm{e}} M_1{:}\tau{\to}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho_1 \;\&$
$B, x{:}\rho_2 \vdash_{\mathrm{E}} M_2{:}\tau \;\&\; B \vdash_{\mathrm{E}} N{:}\rho_2] \;\Rightarrow$ ($\rho = \rho_1{\cap}\rho_2$ & *2.5 (v)* & *2.5 (vii)*)
$\exists\, \rho\; [B, x{:}\rho \vdash_{\mathrm{e}} M_1M_2{:}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho].$ ■

Theorem 2.10 $M =_\beta N \;\Rightarrow\; (B \vdash_{\mathrm{E}} M{:}\sigma \iff B \vdash_{\mathrm{E}} N{:}\sigma)$*, so the following rule is an admissible rule in* $\vdash_{\mathrm{E}}$*:*

$$(=_\beta):\qquad \frac{M{:}\sigma \qquad M =_\beta N}{N{:}\sigma}$$

Proof: By induction on the definition of $=_\beta$. The only part that needs attention is that of a redex, $B \vdash_{\mathrm{E}} (\lambda x.M)N{:}\sigma \iff B \vdash_{\mathrm{E}} M[N/x]{:}\sigma$, where $\sigma \in \mathcal{T}_{\mathrm{s}}$; all other cases follow by straightforward induction. To conclude, notice that, if $B \vdash_{\mathrm{E}} (\lambda x.M)N{:}\sigma$, then, by Lemma *2.5 (ii)* & *(iii)*, $\exists\, \rho$ $[B, x{:}\rho \vdash_{\mathrm{E}} M{:}\sigma \;\&\; B \vdash_{\mathrm{E}} N{:}\rho]$. The result follows then by applying Lemma *2.9*.

3 Approximation and normalization results

In [34] an approximation theorem is proved, that formulates the relation between the types assignable to a term and those assignable to its approximants, as defined in [37] (see Definition *3.1.1* below).

Property 3.1 $B \vdash_{\cap} M{:}\sigma$ *if and only if there exists* $A \in \mathcal{A}(M)$ *such that* $B \vdash_{\cap} A{:}\sigma$. ■

In this section, an 'essential' variant of this property will be proved; for every M, B and σ such that $B \vdash_{\mathrm{E}} M{:}\sigma$, there is an $A \in \mathcal{A}(M)$ such that $B \vdash_{\mathrm{E}} A{:}\sigma$. In [34] 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 [9] and [6]. It requires a rather difficult notion of length of a derivation to show that this process terminates.) In this paper, this result will be proved using the computability technique, following Tait [36], as was done in [11], and [17].

With this result, it can be shown that the BCD-system is conservative over the essential system (Theorem *4.1.5*), and prove that the set of all terms having a (head) normal form are typeable in $\vdash_{\mathrm{e}}$ (with a type without ω-occurrences) (Theorem *3.3.2*).

3.1 Approximate normal forms

The notion of approximant was first presented by C. Wadsworth [37] and is defined using the notion of terms in $\lambda\bot$-normal form (like in [5], $\bot$ 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 [5]).

Definition 3.1.1 *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:*

a) $\lambda x.\bot \rightarrow_{\beta\bot} \bot$.

b) $\bot M \rightarrow_{\beta\bot} \bot$.

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 and is inductively defined by:*

a) $\bot \in \mathcal{N}$.

b) If $A \in \mathcal{N}$, $A \neq \bot$*, then* $\lambda x.A \in \mathcal{N}$.

c) If $A_1, \ldots, A_n \in \mathcal{N}$ $(n \geq 0)$*, then* $xA_1 \cdots A_n \in \mathcal{N}$.

The rules of the essential system are generalized to terms containing $\bot$ by allowing for the terms to be elements of $\Lambda\bot$. This implies that, because essential type assignment is almost syntax directed, if $\bot$ occurs in a term M and $B \vdash_{\mathrm{E}} M{:}\sigma$, then either $\sigma = \omega$, or in the derivation for $M{:}\sigma$, $\bot$ appears in the right hand subterm of an application on which the rule ($\rightarrow$E) is used with $n = 0$. Moreover, the terms $\lambda x.\bot$ and $\bot M_1 \cdots M_n$ are typeable by ω only.

Definition 3.1.2 *i) The relation* $\sqsubseteq\, \subseteq (\Lambda\bot)^2$ *is defined by:*

a) $\bot \sqsubseteq M$.

b) $x \sqsubseteq x$.

c) $M \sqsubseteq M' \Rightarrow \lambda x.M \sqsubseteq \lambda x.M'$.

d) $M_1 \sqsubseteq M_1'$ *&* $M_2 \sqsubseteq M_2' \Rightarrow M_1M_2 \sqsubseteq M_1'M_2'$.

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

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

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

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

The following properties of approximants hold:

Lemma 3.1.3 *i) If $A \in \mathcal{A}(xM_1 \ldots M_n)$ and $A' \in \mathcal{A}(N)$, then $AA' \in \mathcal{A}(xM_1 \ldots M_n N)$.*

ii) If $A \in \mathcal{A}(Mz)$ and $z \notin \mathrm{FV}(M)$, then either:

a) $A \equiv A'z$, $z \notin \mathrm{FV}(A)$, and $A' \in \mathcal{A}(M)$, or

b) $\lambda z.A \in \mathcal{A}(M)$.

Proof: Easy. ■

3.2 Approximation result

In this subsection, the approximation theorem will be proved. For reasons of readability, in this subsection $\exists A \in \mathcal{A}(M)\ [B \vdash_{\mathrm{E}} A{:}\sigma]$ will be abbreviated by $\mathcal{A}pp\,(B, M, \sigma)$.

Lemma 3.2.1 *i) $\mathcal{A}pp\,(B, xM_1 \cdots M_n, \sigma \rightarrow \tau)$ & $\mathcal{A}pp\,(B', N, \sigma) \Rightarrow \mathcal{A}pp\,(\Pi\{B, B'\}, xM_1 \cdots M_n N, \tau)$.*

ii) $\mathcal{A}pp\,(B \cup \{z{:}\sigma\}, Mz, \tau)$ & $z \notin FV(M)$ & $\tau \in \mathcal{T}_{\mathrm{s}} \Rightarrow \mathcal{A}pp\,(B, M, \sigma \rightarrow \tau)$.

iii) $\mathcal{A}pp\,(B, \mathrm{C}[M[N/x]], \sigma) \Rightarrow \mathcal{A}pp\,(B, \mathrm{C}[(\lambda x.M)N], \sigma)$.

Proof: *i)* $A \in \mathcal{A}(xM_1 \cdots M_n)$ & $B \vdash_{\mathrm{E}} A{:}\sigma \rightarrow \tau$ & $A' \in \mathcal{A}(N)$ & $B \vdash_{\mathrm{E}} A'{:}\tau \Rightarrow$ (*3.1.3 (i)* & *2.5 (ii)*)

$AA' \in \mathcal{A}(xM_1 \cdots M_n N)$ & $\Pi\{B, B'\} \vdash_{\mathrm{E}} AA'{:}\tau$.

ii) $A \in \mathcal{A}(Mz)$ & $B, z{:}\sigma \vdash_{\mathrm{E}} A{:}\tau$ & $z \notin \mathrm{FV}(M) \Rightarrow$ (*3.1.3 (ii)*)

a) $A \equiv A'z$ & $z \notin \mathrm{FV}(A')$ & $A' \in \mathcal{A}(M)$ & $B, z{:}\sigma \vdash_{\mathrm{e}} A'z{:}\tau \Rightarrow$ (*2.7*)

$A' \in \mathcal{A}(M)$ & $B \vdash_{\mathrm{e}} A'{:}\sigma \rightarrow \tau$.

b) $\lambda z.A \in \mathcal{A}(M)$ & $B, z{:}\sigma \vdash_{\mathrm{e}} A{:}\tau \Rightarrow \lambda z.A \in \mathcal{A}(M)$ & $B \vdash_{\mathrm{e}} \lambda z.A{:}\sigma \rightarrow \tau$.

iii) Since $M =_{\beta} M'$ implies $\mathcal{A}(M) = \mathcal{A}(M')$. ■

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

Definition 3.2.2 *Comp (B, M, ρ) is inductively defined by:*

i) Comp $(B, M, \varphi) \iff B \vdash_{\mathrm{E}} M{:}\varphi$ & $\mathcal{A}pp\,(B, M, \varphi)$.

ii) Comp $(B, M, \sigma \rightarrow \tau) \iff$ (Comp $(B', N, \sigma) \Rightarrow$ Comp $(\Pi\{B, B'\}, MN, \tau)$).

iii) Comp $(B, M, \sigma_1 \cap \cdots \cap \sigma_n) \iff \forall 1 \leq i \leq n$ [Comp (B, M, σ_i)].

Notice that *Comp* (B, M, ω) holds as special case of part *(iii)*.

Lemma 3.2.3 *Take σ and τ such that $\sigma \leq_{\mathrm{E}} \tau$. Then Comp $(B, M, \sigma) \Rightarrow$ Comp (B, M, τ).*

Proof: By straightforward induction on the definition of $\leq_{\mathrm{E}}$. ■

Lemma 3.2.4 *Comp $(B, \mathrm{C}[M[N/x]], \sigma) \Rightarrow$ Comp $(B, \mathrm{C}[(\lambda x.M)N], \sigma)$.*

Proof: By induction on the structure of types. The case that σ is a type-variable follows from Theorem *2.10* and Lemma *3.2.1 (iii)*. ■

Theorem 3.2.5 *i) $B \vdash_{\mathrm{E}} xM_1 \cdots M_n{:}\rho$ & $\mathcal{A}pp\,(B, xM_1 \cdots M_n, \rho) \Rightarrow$ Comp $(B, xM_1 \cdots M_n, \rho)$.*

ii) Comp $(B, M, \rho) \Rightarrow B \vdash_{\mathrm{E}} M{:}\rho$ & $\mathcal{A}pp\,(B, M, \rho)$.

Proof: Simultaneously by induction on the structure of types. The only interesting case is when $\rho = \sigma \to \tau$; the other cases are dealt with by induction.

i) $B \vdash_{\mathrm{E}} xM_1 \cdots M_n{:}\sigma\to\tau$ & $\mathcal{A}pp\,(B, xM_1\cdots M_n, \sigma\to\tau) \Rightarrow$ (IH *(ii)*)
($Comp\,(B', N, \sigma) \Rightarrow B \vdash_{\mathrm{E}} xM_1\cdots M_n{:}\sigma\to\tau$ & $B' \vdash_{\mathrm{E}} N{:}\sigma$ &
$\mathcal{A}pp\,(B, xM_1\cdots M_n, \sigma\to\tau)$ & $\mathcal{A}pp\,(B', N, \sigma)$) $\Rightarrow$ (*2.5 (ii)* & *3.2.1 (i)*)
($Comp\,(B', N, \sigma) \Rightarrow \Pi\{B, B'\} \vdash_{\mathrm{E}} xM_1\cdots M_nN{:}\tau$ &
$\mathcal{A}pp\,(\Pi\{B, B'\}, xM_1\cdots M_nN, \tau)$) $\Rightarrow$ (IH *(i)*)
($Comp\,(B', N, \sigma) \Rightarrow Comp\,(\Pi\{B, B'\}, xM_1\cdots M_nN, \tau)$) $\Rightarrow$ (*3.2.2 (ii)*)
$Comp\,(B, xM_1\cdots M_n, \sigma\to\tau)$.

ii) $Comp\,(B, M, \sigma\to\tau)$ & $z \notin \mathrm{FV}(M) \Rightarrow$ (IH *(i)*)
$Comp\,(B, M, \sigma\to\tau)$ & $Comp\,(\{z{:}\sigma\}, z, \sigma)$ & $z \notin \mathrm{FV}(M) \Rightarrow$ (*3.2.2 (ii)*)
$Comp\,(B \cup \{z{:}\sigma\}, Mz, \tau)$ & $z \notin \mathrm{FV}(M) \Rightarrow$ (IH *(ii)*)
$B \cup \{z{:}\sigma\} \vdash_{\mathrm{E}} Mz{:}\tau$ & $\mathcal{A}pp\,(B \cup \{z{:}\sigma\}, Mz, \sigma)$ & $z \notin \mathrm{FV}(M) \Rightarrow$ (*2.7* & *3.2.1 (ii)*)
$B \vdash_{\mathrm{E}} M{:}\sigma\to\tau$ & $\mathcal{A}pp\,(B, M, \sigma\to\tau)$. ■

Notice that, by part *(ii)*, in particular $Comp\,(\{x{:}\sigma\}, x, \sigma)$, for all x, σ.

Theorem 3.2.6 If $B = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$, $B \vdash_{\mathrm{E}} M{:}\sigma$, and, for every $1 \leq i \leq n$, $Comp\,(B_i, N_i, \mu_i)$, then $Comp\,(\Pi\{B_1, \ldots, B_n\}, M[N_1/x_1, \ldots, N_n/x_n], \sigma)$.

Proof: By induction on the structure of types, of which only the part $\sigma \in \mathcal{T}_{\mathrm{s}}$ is presented. This is shown by induction on the structure of derivations.

($\to$I): Then $M \equiv \lambda y.M'$, $\sigma = \rho\to\tau$, and $B, y{:}\rho \vdash_{\mathrm{E}} M'{:}\tau$.
$B = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$ & $\forall 1 \leq i \leq n\ [Comp\,(B_i, N_i, \mu_i)]$ &
$B, y{:}\rho \vdash_{\mathrm{E}} M'{:}\tau \Rightarrow$ (IH)
($Comp\,(B', N, \rho) \Rightarrow$
$Comp\,(\Pi\{B_1, \ldots, B_n, B'\}, M'[N_1/x_1, \ldots, N_n/x_n, N/y], \tau)) \Rightarrow$ (*3.2.4*)
($Comp\,(B', N, \rho) \Rightarrow$
$Comp\,(\Pi\{B_1, \ldots, B_n, B'\}, (\lambda y.M'[N_1/x_1, \ldots, N_n/x_n])N, \tau)) \Rightarrow$ (*3.2.2 (ii)*)
$Comp\,(\Pi\{B_1, \ldots, B_n\}, (\lambda y.M')[N_1/x_1, \ldots, N_n/x_n], \rho\to\tau)$.

($\to$E): Then $M \equiv M_1M_2$, $B \vdash_{\mathrm{E}} M_1{:}\rho\to\sigma$, and $B \vdash_{\mathrm{E}} M_2{:}\rho$.
$B = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$ & $\forall 1 \leq i \leq n\ [Comp\,(B_i, N_i, \mu_i)]$ &
$B \vdash_{\mathrm{E}} M_1{:}\rho\to\sigma$ & $B \vdash_{\mathrm{E}} M_2{:}\rho \Rightarrow$ (IH)
$Comp\,(\Pi\{B_1, \ldots, B_n\}, M_1[N_1/x_1, \ldots, N_n/x_n], \rho\to\sigma)$ &
$Comp\,(\Pi\{B_1, \ldots, B_n\}, M_2[N_1/x_1, \ldots, N_n/x_n], \rho) \Rightarrow$ (*3.2.2 (ii)*)
$Comp\,(\Pi\{B_1, \ldots, B_n\}, (M_1M_2)[N_1/x_1, \ldots, N_n/x_n], \sigma)$.

($\leq_{\mathrm{E}}$): By Lemma *3.2.3*. ■

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 3.2.7 $B \vdash_{\mathrm{E}} M{:}\sigma \Leftrightarrow \exists A \in \mathcal{A}(M)\ [B \vdash_{\mathrm{E}} A{:}\sigma]$.

Proof: $\Rightarrow$) $B \vdash_{\mathrm{E}} M{:}\sigma \Rightarrow$ (*3.2.6*) $Comp\,(B, M, \sigma) \Rightarrow$ (*3.2.5 (ii)*) $\exists A \in \mathcal{A}(M)\ [B \vdash_{\mathrm{E}} A{:}\sigma]$.

$\Leftarrow$) If $B \vdash_{\mathrm{E}} A{:}\sigma$, then by the remark made after Definition *3.1.1*, $\perp$ appears only in subterms that are typed by ω. Since $\mathrm{A} \in \mathrm{A}(M)$, there is an M' such that $M' =_{\beta} M$ and $A \sqsubseteq M'$. Then, by Lemma *2.5 (viii)*, $B \vdash_{\mathrm{E}} M'{:}\sigma$ and, by Theorem *2.10*, also $B \vdash_{\mathrm{E}} M{:}\sigma$. ■

3.3 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 3.3.1 i) If $B \vdash_{\mathrm{E}} A{:}\sigma$ 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 $B \vdash_{\mathrm{e}} A{:}\sigma$.

Proof: By induction on A.

i) As before, only the part $\sigma \in \mathcal{T}_{\mathrm{s}}$ is shown; only the part $A \equiv xA_1 \cdots A_n$ is of interest. Then, by Lemma *2.5 (ii)* and *(i)*, there are $\sigma_1, \ldots, \sigma_n, \tau_1, \ldots, \tau_n, \tau$, such that $x{:}\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau \in B$, for every $1 \leq i \leq n$, $B \vdash_{\mathrm{E}} A_i{:}\sigma_i$, and $\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau \leq_{\mathrm{E}} \sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \sigma$. So, especially, for every $1 \leq i \leq n$, $\sigma_i \leq_{\mathrm{E}} \tau_i$. By Theorem *2.8*, also for every $1 \leq i \leq n$, $B \vdash_{\mathrm{E}} A_i{:}\tau_i$. Since each τ_i occurs in B, all are ω-free, so by induction each A_i is $\bot$-free. Then also $xA_1 \cdots A_n$ is $\bot$-free.

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

b) $A \equiv xA_1 \cdots A_n$, with $(n \geq 0)$. By induction there are $B_1, \ldots, B_n$ and $\sigma_1, \ldots, \sigma_n$ such that for every $1 \leq i \leq n$, $B_i \vdash_{\mathrm{e}} A_i{:}\sigma_i$, and B_i, σ_i are ω-free. Take σ strict, such that ω does not occur in σ, and $B = \Pi\{B_1, \ldots, B_n, \{x{:}\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \sigma\}\}$. Then $B \vdash_{\mathrm{E}} xA_1 \cdots A_n{:}\sigma$, and B and σ, are ω-free. ■

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

Theorem 3.3.2 i) $\exists B, \sigma\, [B \vdash_{\mathrm{e}} M{:}\sigma \;\&\; B, \sigma$ ω-free$] \iff M$ has a normal form.

ii) $\exists B, \sigma\, [B \vdash_{\mathrm{e}} M{:}\sigma] \iff M$ has a head normal form.

Proof: i) $\Rightarrow$) If $B \vdash_{\mathrm{e}} M{:}\sigma$, then, by Theorem *3.2.7*, $\exists A \in \mathcal{A}(M)\, [B \vdash_{\mathrm{E}} A{:}\sigma]$. Because of Lemma *3.3.1 (i)*, this A is $\bot$-free. By Definition *3.1.1*, 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 *3.3.1 (ii)*, there are ω-free B, σ such that $B \vdash_{\mathrm{e}} M'{:}\sigma$. Then, by Theorem *2.10*, $B \vdash_{\mathrm{e}} M{:}\sigma$.

ii) $\Rightarrow$) If $B \vdash_{\mathrm{e}} M{:}\sigma$, then, by Theorem *3.2.7*, $\exists A \in \mathcal{A}(M)\, [B \vdash_{\mathrm{E}} A{:}\sigma]$. By Definition *3.1.1*, there exists $M' =_\beta M$ such that $A \sqsubseteq M$. Since $\sigma \in \mathcal{T}_{\mathrm{s}}$, $A \not\equiv \bot$, so A is either $\lambda x.A_1$ or $xA_1 \cdots A_n$, with $n \geq 0$. Since $A \sqsubseteq M'$, M' is either $\lambda x.M_1$, or $xM_1 \cdots M_n$. 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 x, $\lambda x.M_1$ or $xM_1 \cdots M_n$, 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 $B, \sigma \in \mathcal{T}_{\mathrm{s}}$ such that $B \vdash_{\mathrm{E}} M_1{:}\sigma$. If $x{:}\tau \in B$, then $B \backslash x \vdash_{\mathrm{e}} \lambda x.M_1{:}\sigma \rightarrow \tau$, otherwise $B \vdash_{\mathrm{e}} \lambda x.M_1{:}\omega \rightarrow \tau$.

2) $M' \equiv xM_1 \cdots M_n$, $(n \geq 0)$. Take $\sigma \in \mathcal{T}_{\mathrm{s}}$, then
$\Pi\{B_1, \ldots, B_n, \{x{:}\omega \rightarrow \cdots \rightarrow \omega \rightarrow \sigma\}\} \vdash_{\mathrm{e}} xM_1 \cdots M_n{:}\sigma$. ■

Corollary 3.3.3 i) $\exists B, \sigma\, [B \vdash_{\mathrm{E}} M{:}\sigma \;\&\; B, \sigma$ ω-free$] \iff M$ has a normal form.

ii) $\exists B, \sigma\, [B \vdash_{\mathrm{E}} M{:}\sigma \;\&\; \sigma \neq \omega] \iff M$ has a head normal form. ■

4 Soundness and completeness of essential type assignment

4.1 The relation between the BCD- and the essential system

The essential system is the nucleus of the BCD-system: in this subsection it will be shown that, for any derivation in the BCD-system, it is possible to find an equivalent derivation in the essential system.

The proof is based on the fact that for every $\sigma \in \mathcal{T}_\cap$ there is a $\sigma^* \in \mathcal{T}_\mathrm{S}$ such that $\sigma \sim \sigma^*$, and the Approximation Theorem *3.2.7*.

Definition 4.1.1 (cf. [22, 1]) *i) For every* $\sigma \in \mathcal{T}_\cap$, $\sigma^* \in \mathcal{T}_\mathrm{S}$ *is inductively defined as follows:*

a) $\varphi^* = \varphi$.

b) $(\sigma \rightarrow \tau)^* = (\sigma^* \rightarrow \tau_1) \cap \cdots \cap (\sigma^* \rightarrow \tau_n)$, *if* $\tau^* = \tau_1 \cap \cdots \cap \tau_n$ $(n \geq 0)$, *each* $\tau_i \in \mathcal{T}_\mathrm{s}$.

c) $(\sigma_1 \cap \cdots \cap \sigma_n)^* = \tau_1^* \cap \cdots \cap \tau_m^*$, *where* $\{\tau_1, \ldots, \tau_m\} = \{\sigma_i \in \{\sigma_1, \ldots, \sigma_n\} \,|\, \sigma_i^* \neq \omega\}$.

ii) $B^* = \{x{:}\sigma^* \mid x{:}\sigma \in B\}$.

Since $\mathcal{T}_\mathrm{S}$ is a proper subset of $\mathcal{T}_\cap$, σ^* is also defined for $\sigma \in \mathcal{T}_\mathrm{S}$. Notice that $\omega^* = \omega$, as a special case of part *(i.c)*, and that $(\sigma \rightarrow \omega)^* = \omega$, as a special case of part *(i.b)*.

Lemma 4.1.2 *i)* [22, 1] *For every* $\sigma \in \mathcal{T}_\cap$, $\sigma \sim \sigma^*$.

ii) $\mathcal{T}_\cap$ *modulo* $\sim$ *is isomorphic to* $\mathcal{T}_\mathrm{S}$ *modulo* $\sim_\mathrm{E}$.

iii) $\sigma \leq \tau \Rightarrow \sigma^* \leq_\mathrm{E} \tau^*$.

iv) $\sigma \in \mathcal{T}_\mathrm{S} \Rightarrow \sigma = \sigma^*$.

Proof: Easy. ■

The proof for the main theorem of this section is achieved by proving first, that for every term in $\mathcal{N}$, typeable in $\vdash_\cap$, a derivation in the essential system can be built for which basis and type in the conclusion are equivalent, and afterwards generalizing this result to arbitrary lambda terms.

Theorem 4.1.3 $B \vdash_\cap A{:}\sigma \Rightarrow B^* \vdash_\mathrm{E} A{:}\sigma^*$.

Proof: By induction on the structure of terms in $\mathcal{N}$, using Lemmas *1.3.3*, *2.5*, and *4.1.2 (iii)*. ■

The relation between the two different notions of type assignment is formulated as follows:

Theorem 4.1.4 $B \vdash_\cap M{:}\sigma \Rightarrow B^* \vdash_\mathrm{E} M{:}\sigma^*$.

Proof: $B \vdash_\cap M{:}\sigma \Rightarrow (3.1)$ $\exists\, A \in \mathcal{A}(M)\ [B \vdash_\cap A{:}\sigma] \Rightarrow (4.1.3)$
$\exists\, A \in \mathcal{A}(M)\ [B^* \vdash_\mathrm{E} A{:}\sigma^*] \Rightarrow (3.2.7)$ $B^* \vdash_\mathrm{E} M{:}\sigma^*$. ■

The BCD system is a conservative extension of the essential system.

Theorem 4.1.5 Conservativity. *Let* $B, \sigma \in \mathcal{T}_\mathrm{S}$. *If* $B \vdash_\cap M{:}\sigma$, *then* $B \vdash_\mathrm{E} M{:}\sigma$.

Proof: $B \vdash_\cap M{:}\sigma \Rightarrow (4.1.4)$ $B^* \vdash_\mathrm{E} M{:}\sigma^* \Rightarrow (4.1.2\,(iv))$ $B \vdash_\mathrm{E} M{:}\sigma$. ■

Obviously, since the essential system is a subsystem of the BCD-system, the implication in the other direction also holds: If $B \vdash_\mathrm{E} M{:}\sigma$, then $B \vdash_\cap M{:}\sigma$.

Also using this last result, it is possible to prove completeness of essential type assignment with respect to the simple type semantics (see Theorem *4.3.7*).

4.2 An essential filter λ-model

As in [6] and [1], a filter λ-model can be constructed. Names will be used to distinguish between the definition of filters in those papers and the one given here.

Definition 4.2.1 *i) A subset d of $\mathcal{T}_S$ is an essential filter if and only if:*

a) $\sigma_1, \ldots, \sigma_n \in d$ $(n \geq 0)$ $\Rightarrow$ $\sigma_1 \cap \cdots \cap \sigma_n \in d$.

b) $\tau \in d$ & $\tau \leq_E \sigma$ $\Rightarrow$ $\sigma \in d$.

ii) If V is a subset of $\mathcal{T}_S$, then $\uparrow_E V$ is the smallest essential filter that contains V, and $\uparrow_E \sigma = \uparrow_E \{\sigma\}$.

iii) $\mathcal{F}_E = \{d \subseteq \mathcal{T}_S \,|\, d$ is an essential filter$\}$. Application on $\mathcal{F}_E$ is defined by:

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

Notice that every strict filter (Definition *1.4.3*) is an essential filter, and that an essential filter is a filter in the sense of Definition *1.3.4*.

If no confusion is possible, the subscript on $\uparrow$ will be omitted. Notice that an essential filter is never empty; because of part *(i.a)*, for all d, $\omega \in d$. Notice also that the application on filters as in Definition *1.3.4* is not useful for $\mathcal{F}_E$, since it would not be well defined. As in [1], application must be forced to yield filters, since in each arrow type scheme $\sigma \rightarrow \tau \in \mathcal{T}_S$, τ is strict. $<\mathcal{F}_E, \subseteq>$ is a cpo and henceforward it will be considered with the corresponding Scott topology.

For essential filters the following properties hold:

Lemma 4.2.2 *i) $\sigma \in \uparrow\tau \iff \tau \leq_E \sigma$.*

ii) $\sigma \in \uparrow\{\tau \,|\, B \vdash_e M{:}\tau\} \iff \sigma \in \{\tau \,|\, B \vdash_E M{:}\tau\}$. (So $\{\sigma \,|\, B \vdash_E M{:}\sigma\} \in \mathcal{F}_E$.)

Proof: Easy. ■

Definition 4.2.3 *Define F: $\mathcal{F}_E \rightarrow [\mathcal{F}_E \rightarrow \mathcal{F}_E]$ and G: $[\mathcal{F}_E \rightarrow \mathcal{F}_E] \rightarrow \mathcal{F}_E$ by:*

i) $F\, d\, e = d \cdot e$.

ii) $G\, f = \uparrow\{\sigma \rightarrow \tau \,|\, \tau \in f(\uparrow\sigma)\}$.

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

Theorem 4.2.4 *$<\mathcal{F}_E, \cdot>$, with F and G as defined in 4.2.3, is a λ-model.*

Proof: By [5].5.4.1 it is sufficient to prove that F∘G = $\mathrm{Id}_{[\mathcal{F}_E \rightarrow \mathcal{F}_E]}$.
F∘G $f\ d = \uparrow\{\mu \mid \exists\, \rho \in d\, [\rho \rightarrow \mu \in \uparrow\{\sigma \rightarrow \tau \mid \tau \in f(\uparrow\sigma)\}]\}$ = (*4.2.2 (i)*)
$\uparrow\{\mu \mid \exists\, \rho \in d\, [\mu \in f(\uparrow\rho)]\} = f(d)$. ■

Remark that between $\mathcal{F}_E$ and $\mathcal{F}_S$ the same relation exists as between $\mathcal{F}_\cap$ and $\mathcal{F}_S$, as discussed after Definition *1.4.3*.

Definition 4.2.5 *Let ξ be a valuation of term variables in $\mathcal{F}_E$.*

i) $[\![M]\!]_\xi$, the interpretation of terms in $\mathcal{F}_E$ via ξ is inductively defined by:

a) $[\![x]\!]_\xi = \xi(x)$.

b) $[\![MN]\!]_\xi = F\, [\![M]\!]_\xi\, [\![N]\!]_\xi$.

c) $[\![\lambda x.M]\!]_\xi = G\, (\lambda\!\!\lambda\, d \in \mathcal{F}_E.[\![M]\!]_{\xi(d/x)})$.

ii) $B_\xi = \{x{:}\sigma \mid \sigma \in \xi(x)\}$.

Theorem 4.2.6 For all M, ξ: $[\![M]\!]_\xi = \{\sigma \mid B_\xi \vdash_{\mathrm{E}} M{:}\sigma\}$.

Proof: By induction on the structure of lambda terms.

i) $[\![x]\!]_\xi = \xi(x)$. Since $\{y{:}\rho \mid \rho \in \xi(y)\} \vdash_{\mathrm{E}} x{:}\sigma \iff \sigma \in \xi(x)$.

ii) $[\![MN]\!]_\xi = \uparrow\{\tau \mid \exists\,\sigma\,[B_\xi \vdash_{\mathrm{E}} N{:}\sigma \;\&\; B_\xi \vdash_{\mathrm{E}} M{:}\sigma\to\tau]\} =$ (*2.5 (ii)* & *(iv)*)
$\uparrow\{\tau \mid B_\xi \vdash_{\mathrm{e}} MN{:}\tau\} =$ (*4.2.2 (ii)*)
$\{\tau \mid B_\xi \vdash_{\mathrm{E}} MN{:}\tau\}$

iii) $[\![\lambda x.M]\!]_\xi =$
$\uparrow\{\sigma\to\tau \mid B_{\xi(\uparrow\sigma/x)} \vdash_{\mathrm{E}} M{:}\tau\} =$ (*2.5 (iv)*)
$\uparrow\{\sigma\to\tau \mid B_{\xi(\uparrow\sigma/x)} \vdash_{\mathrm{e}} M{:}\tau\} =$ ($B'_\xi = B_\xi \backslash x$)
$\uparrow\{\sigma\to\tau \mid B'_\xi \cup \{x{:}\mu \mid \mu \in \uparrow\sigma\} \vdash_{\mathrm{e}} M{:}\tau\} =$ (*4.2.2 (i)*)
$\uparrow\{\sigma\to\tau \mid B'_\xi \cup \{x{:}\sigma\} \vdash_{\mathrm{e}} M{:}\tau\} =$ (*2.5 (iii)*)
$\uparrow\{\sigma\to\tau \mid B'_\xi \vdash_{\mathrm{e}} \lambda x.M{:}\sigma\to\tau\} =$ (*2.5 (vi)*)
$\uparrow\{\sigma\to\tau \mid B_\xi \vdash_{\mathrm{e}} \lambda x.M{:}\sigma\to\tau\} =$ (*2.5 (iii)* & *4.2.2 (ii)*)
$\{\rho \mid B_\xi \vdash_{\mathrm{E}} \lambda x.M{:}\rho\}$. ■

4.3 Soundness and completeness

In this subsection completeness for the $\vdash_{\mathrm{E}}$ system will be proved. This is done in a way very similar to the one used in [6], using the essential filter λ-model as defined in the previous subsection.

Theorem 4.3.1 Soundness. $B \vdash_{\mathrm{E}} M{:}\sigma \Rightarrow B \models_{\mathrm{s}} M{:}\sigma$.

Proof: By induction on the structure of derivations. ■

The method followed in [6] for the proof of completeness of type assignment is to define a simple type interpretation υ that satisfies: for all types σ, $\upsilon(\sigma) = \{d \in \mathcal{F}_\cap \mid \sigma \in d\}$. The approach taken here is to define a type interpretation, and to show that it is simple.

Definition 4.3.2 i) $\nu_0 : \mathcal{T}_{\mathrm{S}} \to \wp(\mathcal{F}_{\mathrm{E}})$ is defined by: $\nu_0(\sigma) = \{d \in \mathcal{F}_{\mathrm{E}} \mid \sigma \in d\}$.

ii) $\xi_B(x) = \{\sigma \in \mathcal{T}_{\mathrm{S}} \mid B \vdash_{\mathrm{E}} x{:}\sigma\}$.

Theorem 4.3.3 i) The map ν_0 is a simple type interpretation.

ii) If $\sigma \leq_{\mathrm{E}} \tau$, then $\nu_0(\sigma) \subseteq \nu_0(\tau)$.

Proof: i) It is sufficient to check the conditions of Definition *1.3.5 (ii)*.

a) $\forall\, e\,[e \in \nu_0(\sigma) \Rightarrow d\cdot e \in \nu_0(\tau)] \iff$ (*4.2.1 (iii)*)
$\forall\, e\,[e \in \nu_0(\sigma) \Rightarrow \uparrow\{\beta \mid \exists\,\alpha \in e\,[\alpha\to\beta \in d]\} \in \nu_0(\tau)] \iff$ (*4.3.2 (i)*)
$\forall\, e\,[\sigma \in e \Rightarrow \tau \in \uparrow\{\beta \mid \exists\,\alpha \in e\,[\alpha\to\beta \in d]\}] \iff$ ($\tau \in \mathcal{T}_{\mathrm{s}}$)
$\forall\, e\,[\sigma \in e \Rightarrow \exists\,\alpha \in e\,[\alpha\to\tau \in d]] \iff$ (for $\Rightarrow$, take $e = \uparrow\sigma$)
$\sigma\to\tau \in d$.

b) Trivial.

ii) Easy. ■

Lemma 4.3.4 i) $B \vdash_{\mathrm{E}} M{:}\sigma \iff B_{\xi_B} \vdash_{\mathrm{E}} M{:}\sigma$.

ii) $\mathcal{F}_{\mathrm{E}}, \xi_B, \nu_0 \models_{\mathrm{s}} B$.

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

ii) $x{:}\sigma \in B \Rightarrow$ (*(i)*) $\sigma \in \{\tau \mid B_{\xi_B} \vdash_{\mathrm{E}} x{:}\tau\} \Rightarrow \sigma \in [\![x]\!]_{\xi_B}$.
So $[\![x]\!]_{\xi_B} \in \{d \in \mathcal{F}_{\mathrm{E}} \mid \sigma \in d\} = \nu_0(\sigma)$. ■

Since the interpretation of terms by their derivable types gives a λ-model, the following corollary is immediate and an alternative proof for Theorem *2.10*.

Corollary 4.3.5 *If $M =_\beta N$ and $B \vdash_{\mathrm{E}} M{:}\sigma$, then $B \vdash_{\mathrm{E}} N{:}\sigma$.*

Proof: Since $\mathcal{F}_{\mathrm{E}}$ is a λ-model, if $M =_\beta N$, then $[\![M]\!]_{B_\xi} = [\![N]\!]_{B_\xi}$;
so $\{\sigma \mid B \vdash_{\mathrm{E}} M{:}\sigma\} = \{\sigma \mid B \vdash_{\mathrm{E}} N{:}\sigma\}$. ■

Notice that because of the way in which $\vdash_{\mathrm{E}}$ is defined, Corollary *4.3.5* also holds if $\vdash_{\mathrm{E}}$ is replaced by $\vdash_{\mathrm{e}}$.

Theorem 4.3.6 Completeness. *Let $\sigma \in \mathcal{T}_{\mathrm{S}}$, then $B \models_{\mathrm{s}} M{:}\sigma \Rightarrow B \vdash_{\mathrm{E}} M{:}\sigma$.*

Proof: $B \models_{\mathrm{s}} M{:}\sigma \Rightarrow$ (*1.3.6 (ii.c.2)*, *4.3.4 (ii)* & *4.3.3*)
$\mathcal{F}_{\mathrm{E}}, \xi_B, \nu_0 \models_{\mathrm{s}} M{:}\sigma \Rightarrow$ (*1.3.6 (i)*)
$[\![M]\!]_{\xi_B} \in \nu_0(\sigma) \Rightarrow$ (*4.3.2 (i)*)
$\sigma \in [\![M]\!]_{\xi_B} \Rightarrow$ (*4.2.6*)
$B_{\xi_B} \vdash_{\mathrm{E}} M{:}\sigma \Rightarrow$ (*4.3.4 (i)*)
$B \vdash_{\mathrm{E}} M{:}\sigma$. ■

Using the relation between the two notions of type assignment that was defined in the previous subsection, soundness and completeness of essential type assignment can also be proved using the result of [6] (Property *1.3.7*).

Theorem 4.3.7 Soundness and completeness of essential type assignment. *Let B and σ contain types in $\mathcal{T}_{\mathrm{S}}$. Then $B \vdash_{\mathrm{E}} M{:}\sigma \iff B \models_{\mathrm{s}} M{:}\sigma$.*

Proof: $\Rightarrow$) $B \vdash_{\mathrm{E}} M{:}\sigma \Rightarrow B \vdash_\cap M{:}\sigma \Rightarrow$ (*1.3.7 (i)*) $B \models_{\mathrm{s}} M{:}\sigma$.
$\Leftarrow$) $B \models_{\mathrm{s}} M{:}\sigma \Rightarrow$ (*1.3.7 (ii)*) $B \vdash_\cap M{:}\sigma \Rightarrow$ (*4.1.5*) $B \vdash_{\mathrm{E}} M{:}\sigma$. ■

5 Systems with principal types

It is well known that Curry's system has the principal type property: M is typeable if and only if there is a pair $\langle P, \pi \rangle$ of basis and type, called the *principal pair for M*, such that:

- $P \vdash M{:}\pi$, and
- for every pair $\langle B, \sigma \rangle$ such that $B \vdash M{:}\sigma$, there exists an operation O (from a specified set of operations) such that $O(\langle P, \pi \rangle) = \langle B, \sigma \rangle$.

The type π is then called the *principal type for M*. For Curry's system the operation O consists entirely of substitutions, i.e. operations that replace type-variables by types. Principal type schemes for Curry's system are defined by J.R. Hindley in [21]. In this paper the author actually proved the existence of principal types for an object in Combinatory Logic, but the same construction can be used for a proof of the principal type property for terms in Lambda Calculus. The basic idea used in [21] is to define a unification-algorithm, that is used to construct the principal type for an application form the principal types deduced for its components. Since substitution is an easy operation, in Curry's system the set

$$\{\langle B, \sigma \rangle \mid B \vdash M{:}\sigma\}$$

can be computed in a simple way from the principal pair for M.

There exist three intersection systems for which the principal type property is proved: a CDV-system in [9], the BCD-system in [34], and the strict system in [4]. The technique used for the proofs of these properties is very different for the one used for Curry's sytem. The principal type scheme for a term is in [9], [34], and [4] studied through the notion of approximant of a term; terms with a finite number of approximants have finite principal type schemes, while terms with a infinite number of approximants have 'infinite' principal type schemes. It should be noted that, using intersection types, also terms without normal form or, in particular, terms that have an 'infinite' normal form have types. Therefore, a functional characterization of these terms, through a principal type, cannot be represented in a finite way.

5.1 The operation of expansion

As mentioned in the introduction of [9], using intersection types different types can be assigned to the same component of a given term. Therefore, the structure of derivation does not follow the syntactic structure of terms, and with the only operation of substitution, for a given term, not all types can be obtained. This difficulty is overcome in [9] by introducing the (context-dependent) operation of expansion.

The definition of expansion is very complicated. It is an operation on types that deals with the replacement of (sub)types by a number of copies of that type. Expansion on types corresponds to the duplication of (sub)derivations: a subderivation in the right-hand side of an ($\to$E)-step is expanded by copying. In this process, the types that occur in the subderivation are also copied: the types in the conclusion and in the basis of the subderivation will be instantiated into a number of copies.

Suppose the following is a correct derivation:

$$\frac{x{:}\sigma\to\tau \qquad \begin{array}{c} B \\ \bigtriangledown \\ N{:}\sigma \end{array}}{xN{:}\tau}$$

then, in general, the expansion that replaces σ by $\sigma_1\cap\cdots\cap\sigma_n$ creates the following derivation:

$$\frac{x{:}\sigma_1\cap\cdots\cap\sigma_n\to\tau \qquad \begin{array}{c} B_1 \\ \bigtriangledown \\ N{:}\sigma_1 \end{array} \quad \cdots \quad \begin{array}{c} B_n \\ \bigtriangledown \\ N{:}\sigma_n \end{array}}{xN{:}\tau}$$

Suppose that μ is a subtype of σ that is expanded into n copies. If $\rho\to\mu$ is also a subtype of σ, then just replacing μ by an intersection of copies of μ, would generate $\rho\to(\mu_1\cap\cdots\cap\mu_n)$. This is a not a legal type in [9] and [4]. Defining an operation of expansion by saying that it should expand the subtype $\rho\to\mu$ into the type $(\rho\to\mu_1)\cap\cdots\cap(\rho\to\mu_n)$ – which is by definition of the relation $\leq$ a type equivalent to $\rho\to(\mu_1\cap\cdots\cap\mu_n)$ – would give an expansion that is sound, but not sufficient. The subtype $\rho\to\mu$ will, therefore, be expanded into $(\rho_1\to\mu_1)\cap\cdots\cap(\rho_n\to\mu_n)$, where the $\rho_1,\ldots,\rho_n$ are copies of ρ. This means that also all other occurrences of ρ should be expanded into $\rho_1\cap\cdots\cap\rho_n$, with possibly the same effect on other types.

Apparently, the expansion of μ can have a more than local effect on σ. Therefore, the expansion of a type is defined in a such a way that, before the replacement of types by intersections, all sub-types are collected that are affected by the expansion of μ. Then types are traversed top down, and types are replaced if they end with one of the sub-types found.

5.2 The system of [9]

In [9] principal type schemes are defined for a type assignment system that is like the restricted one from [10], but uses the set of types of the unrestricted system. The reason to not use the normalized types as well is the fact that ω is treated as a type constant; since ω can be substituted for φ in $\sigma{\rightarrow}\varphi$, also $\sigma{\rightarrow}\omega$ is considered a type. The principal type property is achieved by defining principal pairs of basis and type for terms in $\lambda\bot$-normal form, specifying the operations of expansion and substitution, proved sufficient to generate all possible pairs for those terms from their principal pair, and generalizing this result to arbitrary lambda terms. This technique is the same as for example used in [34], [4], and Section *6* of this paper.

The set of ground pairs for a term $A{\in}\mathcal{N}$, as defined in [9], is proved to be complete for A, in the sense that all other pairs for A can be generated from a ground pair for A. Ground pairs are those that express the essential structure of a derivation, and types in it are as general as possible with respect to substitutions.

The proof of the principal type property is obtained by first proving the following:

- If $B \vdash A{:}\sigma$ with $A{\in}\mathcal{N}$, then there is a substitution *Sub* and a ground pair $\langle B', \sigma' \rangle$ for A such that *Sub* $(\langle B, \sigma \rangle) = \langle B', \sigma' \rangle$.
- If $\langle B, \sigma \rangle$ is a ground pair for $A{\in}\mathcal{N}$ and $\langle B', \sigma' \rangle$ can be obtained from $\langle B, \sigma \rangle$ by an expansion, then $\langle B', \sigma' \rangle$ is a ground pair for A.
- For all $A{\in}\mathcal{N}$, every ground pair for A is complete for A.

In the construction of principal pairs for lambda terms, first for every $A \in \mathcal{N}$ a particular pair *Pp* (A) is chosen of basis P and type π, called the *principal basis scheme* and *principal type scheme* of A, respectively (see Definition *6.1.5*). This pair is called the *principal pair* of A.

The proof is completed by proving:

- *Pp* (A) is a ground pair for A.
- $B \vdash M{:}\sigma$ if and only if there exists $A{\in}\mathcal{A}(M)$ such that $B \vdash A{:}\sigma$.
- $\{$*Pp* $(A) \mid A{\in}\mathcal{A}(M)\}$ is complete for M.

For an example of a ground pair, and its relation to the operation of expansion, take the pair $\langle \emptyset, (\omega{\rightarrow}(\varphi_0{\rightarrow}\varphi_0){\rightarrow}\varphi_1){\rightarrow}\varphi_1 \rangle$, which is the principal pair of $(\lambda x.x\bot(\lambda y.y))$.

$$\dfrac{\dfrac{\dfrac{[x : \omega{\rightarrow}(\varphi_0{\rightarrow}\varphi_0){\rightarrow}\varphi_1]}{x\bot : (\varphi_0{\rightarrow}\varphi_0){\rightarrow}\varphi_1} \qquad \dfrac{[y : \varphi_0]}{\lambda y.y : \varphi_0{\rightarrow}\varphi_0}}{x\bot(\lambda y.y) : \varphi_1}}{\lambda x.x\bot(\lambda y.y) : (\omega{\rightarrow}(\varphi_0{\rightarrow}\varphi_0){\rightarrow}\varphi_1){\rightarrow}\varphi_1}$$

Let *Exp* be the expansion that copies the subderivation for $\lambda y.y{:}\varphi_0{\rightarrow}\varphi_0$, then

Exp $(\langle \emptyset, (\omega{\rightarrow}(\varphi_0{\rightarrow}\varphi_0){\rightarrow}\varphi_1){\rightarrow}\varphi_1 \rangle) =$

$$\langle \emptyset, (\omega{\rightarrow}(\varphi_2{\rightarrow}\varphi_2){\cap}(\varphi_3{\rightarrow}\varphi_3){\rightarrow}\varphi_1){\rightarrow}\varphi_1 \rangle,$$

which is, by the results mentioned above, a ground pair for $(\lambda x.x\bot(\lambda y.y))$.

$$\dfrac{\dfrac{\dfrac{[x:\omega\to(\varphi_2\to\varphi_2)\cap(\varphi_3\to\varphi_3)\to\varphi_1]}{x\bot:(\varphi_2\to\varphi_2)\cap(\varphi_3\to\varphi_3)\to\varphi_1} \quad \dfrac{[y:\varphi_2]}{\lambda y.y:\varphi_2\to\varphi_2} \quad \dfrac{[y:\varphi_3]}{\lambda y.y:\varphi_3\to\varphi_3}}{x\bot(\lambda y.y):\varphi_1}}{\lambda x.x\bot(\lambda y.y):(\omega\to(\varphi_2\to\varphi_2)\cap(\varphi_3\to\varphi_3)\to\varphi_1)\to\varphi_1}$$

(See Definition *6.2.5*.)

5.3 The system of [34]

For the system as defined in [6], principal type schemes can be defined as in [34]. There three operations are provided – substitution, expansion, and rise – that are sound and sufficient to generate all suitable pairs for a term M from its principal pair.

In this paper, all constructions and definitions are made modulo the equivalence relation $\sim$. In fact, the complexity inserted in the type language of [6] by allowing for intersection types on the right of the arrow type constructor, disturbs greatly the accessibility of this paper. As shown in Section *6*, the results of [34] can also be obtained for the essential system presented here, that is equally powerful.

The first operation defined is substitution, that is defined without restriction: the type that is to be substituted can be every element of $\mathcal{T}_\cap$. Next, the operation of expansion is defined, which is a generalization of the notion of expansion defined in [9]. Both substitution and expansions are in the natural way extended to operations on bases and pairs. The third operation defined (on pairs) is the operation of rise: it consists of adding applications of the derivation rule $(\leq)$ to a derivation. All defined operations are sound in the following sense:

- (Soundness) Let for $A\in\mathcal{N}$, B,σ be such that $B\vdash_\cap A{:}\sigma$, O be an operation of substitution, expansion or rise, and $O(\langle B,\sigma\rangle) = \langle B',\sigma'\rangle$. Then $B'\vdash_\cap A{:}\sigma'$. ■

Linear chains of operations are defined as sequences of operations that start with a number of expansions, followed by a number of substitutions, and that end with *one* rise. (In [34], linear chains are defined as those sequences of operations that start with a number of expansions, followed by a number of substitutions or rises; both are allowed. This definition is not complete, in the sense that the fact that the chain ends with one rise is essential in the proof for completeness.) As in [9], principal pairs are defined for terms in $\lambda\bot$-normal form. The proof of the principal type property is completed by, using the above approximation theorem, proving first that, when $\mathcal{A}(M)$ is finite, then $\{Pp\,(A) \mid A\in\mathcal{A}(M)\}$ has a maximal element (see Theorem *6.3.4* and Definition *6.3.5*). The following is proved:

*Property 5.3.1 i) (*Completeness*) Let $A\in\mathcal{N}$. For any pair $\langle B,\sigma\rangle$ such that $B\vdash_\cap A{:}\sigma$ there exists a linear chain C such that $C\,(Pp\,(A)) = \langle B,\sigma\rangle$.*

ii) Let $B\vdash_\cap M{:}\sigma$.

a) $\mathcal{A}(M)$ is finite. Then $\{Pp\,(A) \mid A\in\mathcal{A}(M)\}$ has a maximal element, say $\langle P,\pi\rangle$. Then there exists a chain C, such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$.

b) $\mathcal{A}(M)$ is infinite. Then there exist a pair $\langle P,\pi\rangle\in\{Pp\,(A) \mid A\in\mathcal{A}(M)\}$ and a chain C, such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$. ■

5.4 The system of [4]

The proof of the principal type property for the strict system as presented in [4] is achieved in a way similar to, but significantly different from, the two techniques sketched above. In that paper, three operations on pairs of basis and types are defined: substitution, expansion, and lifting. The operation of lifting resembles the operation of rise as defined in [34], the operation of substitution is a modification of the one normally used, and the operation of expansion coincides with the one given in [9, 34].

In order to prove that the operations defined are sufficient, three subsets of the set of all pairs of basis and type are defined, namely: principal pairs, ground pairs, and primitive pairs. (The definition of ground pairs coincides with the one given in [9].) In that paper is shown that these form a true hierarchy, that the set of ground pairs for a term is closed under the operation of expansion, that the set of primitive pairs is closed under the operation of lifting, and that the set of pairs is closed for substitution.

The main result of that paper is reached by showing that the three operations defined are complete: if $\langle B,\sigma\rangle$ is a suitable pair for a term A in $\lambda\bot$-normal form, and $\langle P,\pi\rangle$ is the principal pair for A, then there are a sequence of operations of expansion *Exp*, an operation of lifting *Lift*, and a substitution *Sub*, such that

$$\langle B,\sigma\rangle = \mathit{Sub}\,(\mathit{Lift}\,(\overrightarrow{\mathit{Exp}}\,(\langle P,\pi\rangle))).$$

Finally, this result is generalized to arbitrary lambda terms.

Because of technical reasons, substitution is in [4] defined as $(\varphi \mapsto \alpha)$, where φ is a type-variable and $\alpha \in \mathcal{T}_{\rm s} \cup \{\omega\}$, so it can also replace type-variables by the type constant ω (this is not needed in the proofs of Section *6*). Although substitution is normally defined on types as the operation that replaces type-variables by types, for strict types this definition would not be correct. For example, the replacement of φ by ω would transform $\sigma{\rightarrow}\varphi$ (or $\sigma{\cap}\varphi$) into $\sigma{\rightarrow}\omega$ ($\sigma{\cap}\omega$), which is not a strict type. Therefore, for strict types substitution is not defined as an operation that replaces type-variables by types, but as a mapping from types to types, that, in a certain sense, ‘normalizes while substituting’.

The operation of expansion, as defined in [4], corresponds to the one given in [9] and is a simplified version of the one defined in [34]. A difference is that in those definitions subtypes are collected, whereas the definition of expansion in [4] (see Definition *6.2.5*) collects type-variables.

Recall Definition *2.1*. Observe that strict type assignment is not closed for the relation $\leq_{\rm E}$, so the following does not hold:

$$\textit{If } B \vdash_{\rm S} M{:}\sigma \textit{ and } \sigma \leq_{\rm E} \tau, \textit{ then } B \vdash_{\rm S} M{:}\sigma.$$

As a counter example, take $\{x{:}\sigma{\rightarrow}\sigma\} \vdash_{\rm s} x{:}\sigma{\rightarrow}\sigma$. Notice that $\sigma{\rightarrow}\sigma \leq_{\rm E} \sigma{\cap}\tau{\rightarrow}\sigma$, but it is impossible to derive $\{x{:}\sigma{\rightarrow}\sigma\} \vdash_{\rm S} x{:}\sigma{\cap}\tau{\rightarrow}\sigma$.

The operation of lifting as defined in [4] (see Definition *6.2.16*) is based on the relation $\leq_{\rm E}$, in the same way as the operation of rise is based on $\leq$. As shown there, and illustrated above, that operation is not sound on all pairs $\langle B,\sigma\rangle$. (In fact, as argued in [4], it is impossible to formulate an operation that performs the desired lifting and is sound on all pairs.) The reason for this is that introducing a derivation rule, allowed on all terms, using the relation $\leq_{\rm E}$, corresponds to a η-reduction step (see Theorem *2.8*), and the strict system is not closed for η-reduction. Since strict type assignment is *not* closed for $\leq_{\rm E}$, and the operation of lifting applies $\leq_{\rm E}$ to a derivation, it is clear that a conflict arises.

However, in [4] it is shown that the operation defined there is sound on primitive pairs. The definition for primitive pairs is based on the definition of ground pairs as given in [9]. The main difference between ground pairs and primitive pairs is that in a primitive pair a predicate for a term-variable (bound or free) is not the smallest type needed, but can contain some additional, irrelevant types. The problem mentioned above is then solved by allowing liftings only on primitive pairs for terms.

The result of [4] follows from:

- Every principal pair is a ground pair.

- – For every expansion *Exp*, if $\langle B,\sigma\rangle$ is a ground pair for A and $\mathit{Exp}\,(\langle B,\sigma\rangle) = \langle B',\sigma'\rangle$, then $\langle B',\sigma'\rangle$ is a ground pair for A.

  – If $B \vdash_{\mathrm{S}} A{:}\sigma$ and $\mathit{Exp}\,(\langle B,\sigma\rangle) = \langle B',\sigma'\rangle$, then $B' \vdash_{\mathrm{S}} A{:}\sigma'$. ■

- Every ground pair is a primitive pair.
- For all $A \in \mathcal{N}$, liftings *Lift*: if $\langle B,\sigma\rangle$ is a primitive pair for A, then $\mathit{Lift}\,(\langle B,\sigma\rangle)$ is a primitive pair for A. ■
- Every primitive pair is a (normal) pair.
- If $B \vdash_{\mathrm{S}} A{:}\sigma$, then for every substitution *Sub*: if $\mathit{Sub}\,(\langle B,\sigma\rangle) = \langle B,\sigma'\rangle$, then $B' \vdash_{\mathrm{S}} A{:}\sigma'$. ■

Although lifting is not sound on all pairs, using the results mentioned above it is possible to prove that the three operations defined in [4] are sufficient (complete): for every pair $\langle B,\sigma\rangle$ and $A \in \mathcal{N}$, if $B \vdash_{\mathrm{S}} A{:}\sigma$, then there exists a number of expansion, one lifting, and a substitution, such that $\langle B,\sigma\rangle$ can be obtained from $\mathit{Pp}\,(A)$ by performing these operations in sequence. As in [34], this result is then generalized to arbitrary lambda terms (see Property *5.3.1* and Theorem *6.3.6*).

6 Principal type property for the essential system

Using a technique different from those discussed above, in this section the proof for the principal type property of the essential system will be given. For each lambda term the principal pair (of basis and type) will be defined. Four operations on pairs of basis and types will be defined, namely expansion, covering, substitution, and lifting, that are correct and sufficient to generate all derivable pairs for lambda terms in the essential system.

A relevant restriction of the essential system will be presented ($\vdash_{\mathrm{R}}$), that is close to the system defined in Definition *1.2.3*. For this system, the principal type property will be proved, using a technique different from the one used in [9]. In fact, it will be shown that, if $B \vdash_{\mathrm{R}} M{:}\sigma$ and $\langle P,\pi\rangle$ is the principal pair for M, then there is a chain C of operations, consisting of expansions, one covering, and one substitution, such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$. Using this result, the principal type property for the essential system will be proved.

In [4], the main problem to solve was to find an operation of lifting that was able to take the special role of the relation $\leq_{\mathrm{S}}$ into account. As argued in Subsection *5.4*, for the strict system there exist no operation of lifting that is sound on all pairs. Since the essential system is more liberal than the strict one, in the sense that the essential system is closed for the relation $\leq_{\mathrm{E}}$, the operation of lifting as defined in [4] is a sound operation for the essential system (see Theorem *6.2.17*). It is then easy to show that, with just the operations as defined in [4], the principal type property holds for the essential system.

In this subsection a different proof will be presented, that follows a new approach. The most significant difference between proofs for the principal type property made in other papers and the one presented here, is that, in a certain sense, the operations presented in this section are more elegant. In [34], there is an overlap between operations; for example, intersections can be introduced by expansions as well as by substitutions and rise. Also, in [4] the step from the pair $\langle B,\sigma\rangle$ to $\langle B,\omega\rangle$ can be made using a lifting as well as a substitution. The operations of expansion, covering, and substitution as defined in this paper are ‘orthogonal’ in that sense; no kind of operation can be mimicked by another kind of operation.

The difference between the operations specified in [4] and this paper lie in the fact that here the operation of substitution has been changed, in a subtle, natural, but drastic way: a substitution can no longer replace a type-variable by ω. In the papers discussed above that possibility existed and, especially in [9] and [4], caused inconvenience, since there a 'normalization-after-substitution' was called for, explicitly defined in [9], and part of the definition of substitution in [4]. The approach of this paper will be to allow of only substitutions of type variables by strict types, and to introduce a separate operation of *covering*, that deals with the assignment of ω to subterms.

6.1 Relevant intersection type assignment

The next definition presents a restricted variant of the essential system, that is similar to that of Definition *1.2.3*, and is the system used in [19]. Since bases play a more significant role in this system, the presentation of the derivation rules differs from the one used above.

Definition 6.1.1 i) Relevant intersection type assignment and relevant intersection derivations are defined by the following natural deduction system (where all types displayed are in $\mathcal{T}_s$, except for σ in rules ($\rightarrow$E) and ($\rightarrow$I)):

$$\text{(Ax)}: \quad \{x{:}\sigma\} \vdash_R x{:}\sigma \qquad\qquad (\rightarrow E): \quad \frac{B_1 \vdash_R M{:}\sigma\rightarrow\tau \qquad B_2 \vdash_R N{:}\sigma}{\Pi\{B_1,B_2\} \vdash_R MN{:}\tau}$$

$$(\cap I): \quad \frac{B_1 \vdash_R M{:}\sigma_1 \qquad \ldots \qquad B_n \vdash_R M{:}\sigma_n}{\Pi\{B_1,\ldots,B_n\} \vdash_R M{:}\sigma_1\cap\cdots\cap\sigma_n} \;(n \geq 0)$$

$$(\rightarrow I): \quad \frac{B, x{:}\sigma \vdash_R M{:}\tau}{B \vdash_R \lambda x.M{:}\sigma\rightarrow\tau} \qquad \frac{B \vdash_R M{:}\tau}{B \vdash_R \lambda x.M{:}\omega\rightarrow\tau}\;(a)$$

(a) If x does not occur in B.

ii) $\vdash_R$ *is defined by: $B \vdash_R M{:}\sigma$ if and only if there is a relevant derivation that has $B \vdash_R M{:}\sigma$ as conclusion.*

Notice that, by rule ($\cap$I), $\emptyset \vdash_R M{:}\omega$, for all terms M. Notice moreover, that this system is indeed *relevant*, in the sense that only those statements occur in bases that are actually used for the derived statement.

For terms in $\mathcal{N}$, the relation between the essential and the relevant system is formulated by:

Lemma 6.1.2 If A is in $\lambda\bot$-normal form and $B \vdash_E A{:}\sigma$, then there are B',σ' such that $B' \vdash_R A{:}\sigma'$, $\sigma' \leq_E \sigma$ and $B \leq_E B'$.

Proof: By induction on the structure of terms in $\mathcal{N}$.

i) $B \vdash \bot{:}\omega$, then $B' = \emptyset$, and $\sigma' = \omega$.

ii) $B \vdash_E \lambda x.A'{:}\alpha\rightarrow\beta$, with $A' \neq \bot$, then $B \cup \{x{:}\alpha\} \vdash_E A'{:}\beta$. By induction, there are B', ρ', β' such that $\beta' \leq_E \beta$, $B \cup \{x{:}\alpha\} \leq_E B' \cup \{x{:}\alpha'\}$, and $B' \cup \{x{:}\alpha'\} \vdash_R A'{:}\beta'$. Then also $\alpha'\rightarrow\beta' \leq_E \alpha\rightarrow\beta$, $B \leq_E B'$, and $B' \vdash_R \lambda x.A'{:}\alpha'\rightarrow\beta'$.

iii) $B \vdash_E xA_1\cdots A_n{:}\sigma$, with $n \geq 0$. Then there are $\sigma_1,\ldots,\sigma_n$ such that, for $1 \leq i \leq n$, $B \vdash_E A_i{:}\sigma_i$, and $B \vdash_E x{:}\sigma_1\rightarrow\cdots\rightarrow\sigma_n\rightarrow\sigma$. By induction, for every $1 \leq i \leq n$, there are B_i, σ_i', such that $B_i \vdash_R A_i{:}\sigma_i'$, $\sigma_i' \leq_E \sigma_i$, and $B \leq_E B_i$. Then also $B \leq_E \Pi\{B_1,\ldots,B_n,\{x{:}\sigma_1'\rightarrow\cdots\rightarrow\sigma_n'\rightarrow\sigma\}\}$, and $\Pi\{B_1,\ldots,B_n,\{x{:}\sigma_1'\rightarrow\cdots\rightarrow\sigma_n'\rightarrow\sigma\}\} \vdash_R xA_1\cdots A_n{:}\sigma$. ■

Using the same technique as in Subsection *3.2*, the following theorem can be proved.

Theorem 6.1.3 $B \vdash_{\mathrm{R}} M{:}\sigma \iff \exists\, A \in \mathcal{A}(M)\ [B \vdash_{\mathrm{R}} A{:}\sigma]$. ■

Using this approximation result for the relevant system, the following becomes easy.

Theorem 6.1.4 If $B \vdash_{\mathrm{E}} M{:}\sigma$, then there are B', σ' such that $B' \vdash_{\mathrm{R}} M{:}\sigma'$, $\sigma' \leq_{\mathrm{E}} \sigma$ and $B \leq_{\mathrm{E}} B'$.

Proof: If $B \vdash_{\mathrm{E}} M{:}\sigma$ then, by Theorem *3.2.7*, there is an $A \in \mathcal{A}(M)$ such that $B \vdash_{\mathrm{E}} A{:}\sigma$. By Lemma *6.1.2*, there are B', σ' such that $B' \vdash_{\mathrm{R}} A{:}\sigma'$, $\sigma' \leq_{\mathrm{E}} \sigma$ and $B \leq_{\mathrm{E}} B'$. Then, by Theorem *6.1.3*, $B' \vdash_{\mathrm{R}} M{:}\sigma'$. ■

In the construction of principal pairs for lambda terms, first, for any $A \in \mathcal{N}$, a particular pair $Pp\,(A)$ of basis P and type π is chosen, which will be called respectively the *principal basis scheme* and *principal type scheme* of A. Principal pairs for the relevant, and the essential system are defined by:

Definition 6.1.5 i) Let $A \in \mathcal{N}$. $Pp\,(A)$, the principal pair of A, is defined by:

a) $Pp\,(\bot) = \langle \emptyset, \omega \rangle$.

b) $Pp\,(x) = \langle \{x{:}\varphi\}, \varphi \rangle$.

c) If $A \neq \bot$, and $Pp\,(A) = \langle P, \pi \rangle$, then:

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

2) Otherwise $Pp\,(\lambda x.A) = \langle P, \omega \rightarrow \pi \rangle$.

d) If for $1 \leq i \leq n$, $Pp\,(A_i) = \langle P_i, \pi_i \rangle$ (disjoint in pairs), then

$$Pp\,(xA_1 \cdots A_n) = \langle \Pi\{P_1, \ldots, P_n, \{x{:}\pi_1 \rightarrow \cdots \rightarrow \pi_n \rightarrow \varphi\}\}, \varphi \rangle,$$

where φ is a type-variable that does not occur in $Pp\,(A_i)$, for $1 \leq i \leq n$.

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

The following result is almost immediate:

Lemma 6.1.6 If $Pp\,(A) = \langle P, \pi \rangle$, then $P \vdash_{\mathrm{R}} A{:}\pi$ and $P \vdash_{\mathrm{E}} A{:}\pi$.

Proof: Easy. ■

The notion of principal pairs for terms in $\mathcal{N}$ will be generalized to arbitrary lambda terms in Definition *6.3.5*.

The principal pairs in the systems as presented in [9], [34], and [4] are exactly the same. Since the essential type assignment system is a sub-system of the BCD-system (in the sense that if $B \vdash_{\mathrm{E}} M{:}\sigma$, then also $B \vdash_{\cap} M{:}\sigma$, but not vice-versa), and it is a super-system for both the strict and the restricted CDV-system, it not surprising that the principal pairs in the essential system turn out to be exactly the same as the principal pairs in all the other systems.

6.2 Operations

Substitution is normally defined on types as the operation that replaces type-variables by types, without restriction. The notion of substitution defined here replaces type-variables by *strict* types only. Although this is a severe restriction with regard to the usual approach, the operation will proven to be sufficient.

Definition 6.2.1 *i) The substitution* $(\varphi \mapsto \alpha) : \mathcal{T}_S \rightarrow \mathcal{T}_S$, *where* φ *is a type-variable and* $\alpha \in \mathcal{T}_s$, *is defined by:*

a) $(\varphi \mapsto \alpha)(\varphi) = \alpha$.

b) $(\varphi \mapsto \alpha)(\varphi') = \varphi'$, *if* $\varphi \not\equiv \varphi'$.

c) $(\varphi \mapsto \alpha)(\sigma \rightarrow \tau) = (\varphi \mapsto \alpha)(\sigma) \rightarrow (\varphi \mapsto \alpha)(\tau)$.

d) $(\varphi \mapsto \alpha)(\sigma_1 \cap \cdots \cap \sigma_n) = (\varphi \mapsto \alpha)(\sigma_1) \cap \cdots \cap (\varphi \mapsto \alpha)(\sigma_n)$.

ii) If Sub_1 *and* Sub_2 *are substitutions, then so is* $Sub_1 \circ Sub_2$, *where* $Sub_1 \circ Sub_2(\sigma) = Sub_1(Sub_2(\sigma))$.

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

iv) $Sub(\langle B, \sigma \rangle) = \langle Sub(B), Sub(\sigma) \rangle$.

The operation of substitution is sound for the relevant system.

Theorem 6.2.2 *If* $B \vdash_R A{:}\sigma$, *then for every substitution Sub: if* $Sub(\langle B, \sigma \rangle) = \langle B', \sigma' \rangle$, *then* $B' \vdash_R A{:}\sigma'$.

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

The following is needed in the proof of Theorem *6.3.3*.

Lemma 6.2.3 *Let* $\tau \in \mathcal{T}_s$ *and Sub be a substitution such that* $Sub(\tau) = \tau'$. *Then:*

i) If $Sub(B \cup \{x{:}\sigma\}) = B' \cup \{x{:}\sigma'\}$, *then* $Sub(\langle B, \sigma \rightarrow \tau \rangle) = \langle B', \sigma' \rightarrow \tau' \rangle$.

ii) If for every $1 \leq i \leq n$, $Sub(\langle B_i, \sigma_i \rangle) = \langle B_i', \sigma_i' \rangle$, *then*

$$Sub(\langle \Pi\{B_1, \ldots, B_n, \{x{:}\sigma_1 \rightarrow \cdots \rightarrow \sigma_n \rightarrow \tau\}\}, \tau \rangle) = \langle \Pi\{B_1', \ldots, B_n', \{x{:}\sigma_1' \rightarrow \cdots \rightarrow \sigma_n' \rightarrow \tau'\}\}, \tau' \rangle.$$

Proof: Immediately by Definition *6.2.1*. ■

The operation of expansion of types defined here corresponds to the notion of expansion as defined in [34] and [4]. A difference between the notions of expansion as defined in [9] and [34] is that in those papers a *set of types involved in the expansion* is created. As in [4], here just type-variables are collected, so the definition of expansion presented here is less complicated.

Definition 6.2.4 *i) If* B *is a basis and* $\sigma \in \mathcal{T}_S$, *then* $\mathcal{T}_{\langle B, \sigma \rangle}$ *is the set of all strict subtypes occurring in the pair* $\langle B, \sigma \rangle$.

ii) The last type-variable of a strict type is defined by:

a) The last type-variable of φ *is* φ.

b) The last type-variable of $\sigma_1 \cap \cdots \cap \sigma_n \rightarrow \sigma$ *is the last type-variable of* σ.

Definition 6.2.5 *For every* $\mu \in \mathcal{T}_s$, $n \geq 2$, *basis* B, *and* $\sigma \in \mathcal{T}_S$, *the quadruple* $\langle \mu, n, B, \sigma \rangle$ *determines an expansion* $Exp_{\langle \mu, n, B, \sigma \rangle} : \mathcal{T}_S \rightarrow \mathcal{T}_S$, *that is constructed as follows.*

i) The set of type-variables $\mathcal{V}_\mu(\langle B, \sigma \rangle)$ *affected by* $Exp_{\langle \mu, n, B, \sigma \rangle}$ *is constructed by:*

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

b) If the last type-variable of $\tau \in \mathcal{T}_{\langle B, \sigma \rangle}$ *is in* $\mathcal{V}_\mu(\langle B, \sigma \rangle)$, *then for all type-variables* φ *that occur in* τ: $\varphi \in \mathcal{V}_\mu(\langle B, \sigma \rangle)$.

ii) Suppose $\mathcal{V}_\mu(\langle B, \sigma \rangle) = \{\varphi_1, \ldots, \varphi_m\}$. *Choose* $m \times n$ *different type-variables* $\varphi_1^1, \ldots, \varphi_1^n, \ldots, \varphi_m^1, \ldots, \varphi_m^n$, *such that each* φ_j^i *does not occur in* $\langle B, \sigma \rangle$, *for* $1 \leq i \leq n$ *and* $1 \leq j \leq m$. *Let, for* $1 \leq i \leq n$, Sub_i *be such that* $Sub_i(\varphi_j) = \varphi_j^i$, *for* $1 \leq j \leq m$.

iii) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(\tau)$ is obtained by traversing τ top-down and replacing every subtype α by $Sub_1\,(\alpha)\cap\cdots\cap Sub_n\,(\alpha)$, if the last type-variable of α is in $\mathcal{V}_\mu(\langle B,\sigma\rangle)$, i.e.:*
  a) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(\tau_1\cap\cdots\cap\tau_n) = Exp_{\langle\mu,n,B,\sigma\rangle}\,(\tau_1)\cap\cdots\cap Exp_{\langle\mu,n,B,\sigma\rangle}\,(\tau_n)$.*
  b) *If the last type-variable of τ is in $\mathcal{V}_\mu(\langle B,\sigma\rangle)$, then $Exp_{\langle\mu,n,B,\sigma\rangle}\,(\tau) = Sub_1\,(\tau)\cap\cdots\cap Sub_n\,(\tau)$.*
  c) *Otherwise,*
    1) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(\rho\to\mu) = Exp_{\langle\mu,n,B,\sigma\rangle}\,(\rho)\to Exp_{\langle\mu,n,B,\sigma\rangle}\,(\mu)$.*
    2) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(\varphi) = \varphi$.*

iv) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(B') = \{x{:}Exp_{\langle\mu,n,B,\sigma\rangle}\,(\rho) \mid x{:}\rho\in B'\}$.*

v) *$Exp_{\langle\mu,n,B,\sigma\rangle}\,(\langle B',\sigma'\rangle) = \langle Exp_{\langle\mu,n,B,\sigma\rangle}\,(B'), Exp_{\langle\mu,n,B,\sigma\rangle}\,(\sigma')\rangle$.*

Instead of $Exp_{\langle\mu,n,B,\sigma\rangle}$, the notation $\langle\mu,n,B,\sigma\rangle$ will be used.

Example 6.2.6 *Let μ be the type $(\varphi_1\to\varphi_2)\to(\varphi_3\to\varphi_1)\to\varphi_3\to\varphi_2$, and Exp be the expansion $\langle\varphi_1,2,\emptyset,\mu\rangle$. Then $\mathcal{V}_{\varphi_1}(\langle\emptyset,\mu\rangle) = \{\varphi_1, \varphi_3\}$, and*

$$Exp\,(\mu) = ((\varphi_4\cap\varphi_5)\to\varphi_2)\to((\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5))\to(\varphi_6\cap\varphi_7)\to\varphi_2.$$

Notice that both $\langle\emptyset,\mu\rangle$ and $Exp\,(\langle\emptyset,\mu\rangle)$ are pairs for the term $\lambda xyz.x(yz)$ in $\vdash_{\mathrm{E}}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{x:\varphi_1\to\varphi_2 \qquad \dfrac{y:\varphi_3\to\varphi_1 \qquad z:\varphi_3}{yz:\varphi_1}}{x(yz):\varphi_2}}{\lambda z.x(yz):\varphi_3\to\varphi_2}}{\lambda yz.x(yz):(\varphi_3\to\varphi_1)\to\varphi_3\to\varphi_2}}{\lambda xyz.x(yz):(\varphi_1\to\varphi_2)\to(\varphi_3\to\varphi_1)\to\varphi_3\to\varphi_2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{x:(\varphi_4\cap\varphi_5)\to\varphi_2 \qquad \dfrac{\dfrac{y:(\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5)}{y:\varphi_6\to\varphi_4} \qquad \dfrac{z:\varphi_6\cap\varphi_7}{z:\varphi_6}}{yz:\varphi_4} \qquad \dfrac{\dfrac{y:(\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5)}{y:\varphi_7\to\varphi_5} \qquad \dfrac{z:\varphi_6\cap\varphi_7}{z:\varphi_7}}{yz:\varphi_5}}{x(yz):\varphi_2}}{\lambda z.x(yz):(\varphi_6\cap\varphi_7)\to\varphi_2}}{\lambda yz.x(yz):((\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5))\to(\varphi_6\cap\varphi_7)\to\varphi_2}}{\lambda xyz.x(yz):((\varphi_4\cap\varphi_5)\to\varphi_2)\to((\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5))\to(\varphi_6\cap\varphi_7)\to\varphi_2}$$

For an operation of expansion the following property holds:

Lemma 6.2.7 *Let Exp = $\langle\mu,n,B,\sigma\rangle$ be an expansion. Exp $(\tau) = \tau_1\cap\cdots\cap\tau_n$ with for every $1\leq i\leq n$, τ_i is a trivial variant of τ, or Exp $(\tau)\in\mathcal{T}_{\mathrm{s}}$.*

Proof: Immediately by Definition *6.2.5*. ■

The following lemmas are needed in the proofs of the following theorems. The first states that if the last type-variable of the type in a pair is affected by an expansion, then all type-variables in that pair are affected.

Lemma 6.2.8 Let $B' \vdash_{\mathrm{R}} A{:}\tau$, where $\tau \in \mathcal{T}_{\mathrm{s}}$ with last type-variable φ, and $\langle \mu, n, B, \sigma \rangle$ be an expansion such that $\mathcal{T}_{\langle B',\tau\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle}$. If $\varphi \in \mathcal{V}_{\mu}(\langle B, \sigma\rangle)$, then, for every type-variable φ' that occurs in $\langle B', \tau\rangle$, $\varphi' \in \mathcal{V}_{\mu}(\langle B, \sigma\rangle)$.

Proof: By induction on the structure of elements of $\mathcal{N}$.

i) $A \equiv x$, then $B' = \{x{:}\tau\}$. Since the last type-variable of τ is in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$, and

$$\tau \in \mathcal{T}_{\langle\{x:\tau\},\tau\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle},$$

all type-variables that occur in τ are in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$.

ii) $A \equiv \lambda x.A'$, then $\tau = \alpha{\rightarrow}\beta$, and $B' \cup \{x{:}\alpha\} \vdash_{\mathrm{R}} A'{:}\beta$ (if $\alpha = \omega$, then $B' \cup \{x{:}\alpha\} = B'$). Since the last type-variable of $\alpha{\rightarrow}\beta$ is the last type-variable of β, and

$$\mathcal{T}_{\langle B' \cup \{x:\alpha\},\beta\rangle} \subseteq \mathcal{T}_{\langle B',\alpha\rightarrow\beta\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle},$$

by induction, all type-variables in $\langle B' \cup \{x{:}\alpha\}, \beta\rangle$ are in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$. So all type-variables in $\langle B', \alpha{\rightarrow}\beta\rangle$ are in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$.

iii) $A \equiv xA_1 \cdots A_m$. Then there are $\tau_1, \ldots, \tau_m$, $B_1, \ldots, B_m$, such that for every $1 \leq j \leq m$, $B_j \vdash_{\mathrm{R}} A_j{:}\tau_j$, and $B' = \Pi\{B_1, \ldots, B_m, \{x{:}\tau_1{\rightarrow}\cdots{\rightarrow}\tau_m{\rightarrow}\tau\}\}$. Since the last type-variable of τ is in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$, and

$$\tau_1{\rightarrow}\cdots{\rightarrow}\tau_m{\rightarrow}\tau \in \mathcal{T}_{\langle\Pi\{B_0,\ldots,B_m,\{x:\tau_1\rightarrow\cdots\rightarrow\tau_m\rightarrow\tau\}\},\tau\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle},$$

every type-variable in $\tau_1{\rightarrow}\cdots{\rightarrow}\tau_m{\rightarrow}\tau$ is in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$. If, for $1 \leq j \leq m$, $\tau_j = \tau_j^1 \cap \cdots \cap \tau_j^{k_j}$, then, for every $1 \leq l \leq k_j$, the last type-variable of τ_j^l is in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$, and

$$\mathcal{T}_{\langle B_j,\tau_j^l\rangle} \subseteq \mathcal{T}_{\langle\Pi\{B_0,\ldots,B_m,\{x:\tau_1\rightarrow\cdots\rightarrow\tau_m\rightarrow\tau\}\},\tau\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle},$$

so all type-variables in $\langle B_j, \tau_j^l\rangle$ are in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$, for $1 \leq j \leq m$, $1 \leq l \leq k_j$. So all type-variables in $\langle\Pi\{B_1, \ldots, B_m, \{x{:}\tau_1{\rightarrow}\cdots{\rightarrow}\tau_m{\rightarrow}\tau\}\}, \tau\rangle$ are in $\mathcal{V}_{\mu}(\langle B, \sigma\rangle)$. ■

Lemma 6.2.9 Let $B' \vdash_{\mathrm{R}} A{:}\tau$, where $\tau \in \mathcal{T}_{\mathrm{s}}$, and Exp = $\langle \mu, n, B, \sigma\rangle$ be an expansion such that $\mathcal{T}_{\langle B',\tau\rangle} \subseteq \mathcal{T}_{\langle B,\sigma\rangle}$. Then either there are $B_1, \ldots, B_n$, $\tau_1, \ldots, \tau_n$, such that

$$\mathit{Exp}\,(\langle B', \tau\rangle) = \langle \Pi\{B_1, \ldots, B_n\}, \tau_1 \cap \cdots \cap \tau_n\rangle$$

and, for every $1 \leq i \leq n$, $\langle B_i, \tau_i\rangle$ is a trivial variant of $\langle B', \tau\rangle$, or Exp $(\langle B', \tau\rangle) = \langle B'', \tau'\rangle$, with $\tau' \in \mathcal{T}_{\mathrm{s}}$.

Proof: By Lemmas *6.2.7*, and *6.2.8*. ■

Notice that, in particular, this lemma holds for the case that $\langle B', \tau\rangle = \langle B, \sigma\rangle$.

The following property is needed in the proofs for the fact that expansion is sound (Theorem *6.2.11*), and for completeness of chains of operations (Theorem *6.3.3*).

Property 6.2.10 ([4]) *Let Exp be an expansion, $\sigma \in \mathcal{T}_{\mathrm{s}}$ such that the last type-variable of σ is not affected by Exp, and Exp $(\sigma) = \sigma'$. Then:*

i) Exp $(\langle B \cup \{x{:}\tau\}, \sigma\rangle) = \langle B' \cup \{x{:}\tau'\}, \sigma'\rangle$, if and only if Exp $(\langle B, \tau{\rightarrow}\sigma\rangle) = \langle B', \tau'{\rightarrow}\sigma'\rangle$.

ii) Let Exp $(\langle B_i, \sigma_i\rangle) = \langle B_i', \sigma_i'\rangle$, for every $1 \leq i \leq n$. Then

$$\mathit{Exp}\,(\langle\Pi\{B_1, \ldots, B_n, \{x{:}\sigma_1{\rightarrow}\cdots{\rightarrow}\sigma_n{\rightarrow}\sigma\}\}, \sigma\rangle) = \langle\Pi\{B_1', \ldots, B_n', \{x{:}\sigma_1'{\rightarrow}\cdots{\rightarrow}\sigma_n'{\rightarrow}\sigma'\}\}, \sigma'\rangle.$$

The following theorem states that expansion is sound for relevant type assignment.

Theorem 6.2.11 If $B \vdash_{\mathrm{R}} A{:}\sigma$ and Exp an expansion such that Exp $(\langle B, \sigma\rangle) = \langle B', \sigma'\rangle$, then $B' \vdash_{\mathrm{R}} A{:}\sigma'$.

Proof: By induction on the definition of $\vdash_R$, of which only the part $\sigma \in \mathcal{T}_s$ is shown. By Lemma *6.2.9* either:

i) $\sigma' = \sigma_1 \cap \cdots \cap \sigma_m$, $B' = \Pi\{B_1, \ldots, B_m\}$, and each $\langle B_i, \sigma_i \rangle$ is a trivial variant of $\langle B, \sigma \rangle$ and, therefore, $B_i \vdash_R A{:}\sigma_i$. So, by rule $(\cap I)$, $B' \vdash_R A{:}\sigma'$.

or

ii) $\sigma' \in \mathcal{T}_s$. This part is proved by induction on the structure of elements of $\mathcal{N}$. Notice that the case that $A \equiv \bot$ need not be considered.

a) $A \equiv \lambda x.A'$, $\sigma = \alpha \to \beta$, and $B \cup \{x{:}\alpha\} \vdash_R A'{:}\beta$. Let $\sigma' = \alpha' \to \beta'$. (Notice that, if $\alpha = \omega$, by Definition *6.2.5*, also $\alpha' = \omega$). By Lemma *6.2.10 (i)*, *Exp* $(\langle B \cup \{x{:}\alpha\}, \beta \rangle) = \langle B' \cup \{x{:}\alpha'\}, \beta' \rangle$, and, by induction, $B' \cup \{x{:}\alpha'\} \vdash_R A'{:}\beta'$. Then also $B' \vdash_R \lambda x.A'{:}\alpha' \to \beta'$.

b) $A \equiv x A_1 \cdots A_n$, with $n \geq 0$, $B = \Pi\{B_1, \ldots, B_n, \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma\}\}$, and $B_i \vdash_R A_i{:}\sigma_i$, for $1 \leq i \leq n$. Let, for $1 \leq i \leq n$, *Exp* $(\langle B_i, \sigma_i \rangle) = \langle B'_i, \sigma'_i \rangle$, then, by induction, for every $1 \leq i \leq n$, $B'_i \vdash_R A_j{:}\sigma'_i$. Then, by Lemma *6.2.10 (ii)*,

$$\mathit{Exp}\,(\langle \Pi\{B_1, \ldots, B_m, \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma\}\}, \sigma \rangle) = \langle \Pi\{B'_1, \ldots, B'_m, \{x{:}\sigma'_1 \to \cdots \to \sigma'_n \to \sigma'\}\}, \sigma' \rangle.$$

Then $\Pi\{B'_1, \ldots, B'_m, \{x{:}\sigma'_1 \to \cdots \to \sigma'_n \to \sigma'\}\} \vdash_R x A_1 \cdots A_m{:}\sigma'$. ■

The third operation on pairs defined in this section is the operation of covering. It is, unlike the definition of lifting and rise, not defined on types, but directly on pairs, using the relation $\prec\!\!\prec$ defined on pairs. This relation is inspired by the relation $\sqsubseteq$ on terms in $\mathcal{N}$, and the relation between the principal pairs of two terms that are in that relation (see also Theorem *6.3.4*).

Definition 6.2.12 The relation on pairs $\prec\!\!\prec$ is defined by:

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

ii) $\forall 1 \leq i \leq n \ (n \geq 2)\ [\langle B_i, \sigma_i \rangle \prec\!\!\prec \langle B'_i, \sigma'_i \rangle] \Rightarrow$
$$\langle \Pi\{B_1, \ldots, B_n\}, \sigma_1 \cap \cdots \cap \sigma_n \rangle \prec\!\!\prec \langle \Pi\{B'_1, \ldots, B'_n\}, \sigma'_1 \cap \cdots \cap \sigma'_n \rangle.$$

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

iv) $\forall 1 \leq i \leq n\ [\langle B_i, \sigma_i \rangle \prec\!\!\prec \langle B'_i, \sigma'_i \rangle] \Rightarrow$
$$\langle \Pi\{B_1, \ldots, B_n, \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \sigma\}\}, \sigma \rangle \prec\!\!\prec \langle \Pi\{B'_1, \ldots, B'_n, \{x{:}\sigma'_1 \to \cdots \to \sigma'_n \to \sigma\}\}, \sigma \rangle.$$

Definition 6.2.13 A covering Cov is an operation denoted by a pair of pairs $\prec\langle B_0, \tau_0 \rangle, \langle B_1, \tau_1 \rangle\succ$ such that $\langle B_0, \tau_0 \rangle \prec\!\!\prec \langle B_1, \tau_1 \rangle$, and is defined by:

$$\begin{aligned} \mathit{Cov}\,(\langle B, \sigma \rangle) &= \langle B_1, \tau_1 \rangle, \text{ if } \langle B, \sigma \rangle = \langle B_0, \tau_0 \rangle, \\ &= \langle B, \sigma \rangle, \quad \text{otherwise.} \end{aligned}$$

As mentioned above, the operation of covering defined here is inspired by the relation $\sqsubseteq$ on terms in $\mathcal{N}$, and, in fact, is very close to a notion defined in [27]. In that paper principal typings for the type assignment system as presented in [25] are studied. That system is a combination of the BCD-system and the polymorphic type discipline as presented in [20], and can be seen as an extension of the BCD-system by adding quantification over type-variables.

In [27], for every A in $\lambda\bot$-normal form a relation $\subseteq_A$ is defined on the inductively defined set of pairs $\langle B, \sigma \rangle$ admissible for A (i.e. such that $B \vdash A{:}\sigma$). This relation satisfies:

If $\langle B_1, \sigma_1 \rangle \subseteq_A \langle B_2, \sigma_2 \rangle$, then both pairs are admissible for A.

It is, for example, straightforward to show that $\langle B_1,\sigma_1\rangle \subseteq_A \langle B_2,\sigma_2\rangle$, when restricted to $\vdash_{\rm E}$, implies $\langle B_1,\sigma_1\rangle \prec\!\!\prec \langle B_2,\sigma_2\rangle$. Notice that the structure of a term A is present in the relation $\subseteq_A$, but absent in $\prec\!\!\prec$, although of course Definition *6.2.12* follows in part the syntactic structure of terms in $\mathcal{N}$ (part *(iii)* and *(iv)*).

The operation of covering is not sound for $\vdash_{\rm E}$.

Example 6.2.14 It is easy to check that $\langle\{x{:}\alpha\cap(\alpha\to\alpha)\},\alpha\rangle \prec\!\!\prec \langle\{x{:}\omega\to\alpha\},\alpha\rangle$. *Notice that the first pair is legal since* $\{x{:}\alpha\cap(\alpha\to\alpha)\}\vdash_{\rm E} x{:}\alpha$ *is derivable, but not* $\{x{:}\omega\to\alpha\}\vdash_{\rm E} x{:}\alpha$.

The operation of covering is sound for the relevant system.

Theorem 6.2.15 For every covering $\prec\langle B,\sigma\rangle,\langle B',\sigma'\rangle\succ$, *if* $B\vdash_{\rm R} A{:}\sigma$, *then* $B'\vdash_{\rm R} A{:}\sigma'$.

Proof: By induction on the structure of types.

i) $\sigma' = \omega$, $B' = \emptyset$. Trivial.

ii) $\sigma' = \sigma'_1\cap\cdots\cap\sigma'_n$, $n\geq 2$. Then $B = \Pi\{B_1,\ldots,B_n\}$, $\sigma=\sigma_1\cap\cdots\cap\sigma_n$, and, for every $1\leq i\leq n$, $B_i\vdash_{\rm R} A{:}\sigma_i$. By Definition *6.2.12 (ii)*, $B' = \Pi\{B'_1,\ldots,B'_n\}$, and, for every $1\leq i\leq n$, $\prec\langle B_i,\sigma_i\rangle,\langle B'_i,\sigma'_i\rangle\succ$ is a covering. Then, by induction, $B'_i\vdash_{\rm R} A{:}\sigma'_i$, so also $B'\vdash_{\rm R} A{:}\sigma'_1\cap\cdots\cap\sigma'_n$.

iii) $\sigma'\in\mathcal{T}_{\rm s}$. By induction on A.

a) $A\equiv\lambda x.A'$. If $B\vdash_{\rm R}\lambda x.A'{:}\sigma$, then $\sigma=\rho\to\mu$, $B,x{:}\rho\vdash_{\rm R} A'{:}\mu$, and $\sigma'=\rho'\to\mu'$. Since $\prec\langle B,\rho\to\mu\rangle,\langle B',\rho'\to\mu'\rangle\succ$ is a covering, also $\prec\langle B\cup\{x{:}\rho\},\mu\rangle,\langle B'\cup\{x{:}\rho'\},\mu'\rangle\succ$ is a covering, so, by induction, $B',x{:}\rho'\vdash_{\rm R} A'{:}\mu'$, so $B'\vdash_{\rm R}\lambda x.A'{:}\rho'\to\mu'$.

b) $A\equiv yA_1\cdots A_n$, with $n\geq 0$. Since $B\vdash_{\rm R} yA_1\cdots A_n{:}\sigma$, there are $B_1,\ldots,B_n$, $\sigma_1,\ldots,\sigma_n$, such that $B=\Pi\{B_1,\ldots,B_n,\{x{:}\sigma_1\to\cdots\to\sigma_n\to\sigma\}\}$, and $B_i\vdash_{\rm R} A_i{:}\sigma_i$, for every $1\leq i\leq n$. Then there are $B'_1,\ldots,B'_n$, $\sigma'_1,\ldots,\sigma'_n$, such that for every $1\leq i\leq n$, $\prec\langle B_i,\sigma_i\rangle,\langle B'_i,\sigma'_i\rangle\succ$ is a covering, and $B'=\Pi\{B'_1,\ldots,B'_n,\{x{:}\sigma'_1\to\cdots\to\sigma'_n\to\sigma\}\}$. Then by induction, for $1\leq i\leq n$, $B'_i\vdash_{\rm R} A_i{:}\sigma'_i$, so also $\Pi\{B'_1,\ldots,B'_n,\{x{:}\sigma'_1\to\cdots\to\sigma'_n\to\sigma\}\}\vdash_{\rm R} yA_1\cdots A_n{:}\sigma$. ■

The last operation needed in this paper is that of lifting, as first presented in [4]:

Definition 6.2.16 A lifting Lift is an operation denoted by a pair of pairs $<\langle B_0,\tau_0\rangle,\langle B_1,\tau_1\rangle>$ *such that* $\tau_0\leq_{\rm E}\tau_1$ *and* $B_1\leq_{\rm E} B_0$, *and is defined by:*

i) Lift $(\sigma)=\tau_1$, *if* $\sigma=\tau_0$; *otherwise, Lift* $(\sigma)=\sigma$.

ii) Lift $(B)=B_1$, *if* $B=B_0$; *otherwise, Lift* $(B)=B$.

iii) Lift $(\langle B,\sigma\rangle)=\langle$*Lift* (B), *Lift* $(\sigma)\rangle$.

The operation of lifting is sound for essential type assignment.

Theorem 6.2.17 If $B\vdash_{\rm E} M{:}\sigma$ *and* $<\langle B,\sigma\rangle,\langle B',\sigma'\rangle>$ *is a lifting, then* $B'\vdash_{\rm E} M{:}\sigma'$.

Proof: By definition *6.2.16*, $B'\leq_{\rm E} B$, and $\sigma\leq_{\rm E}\sigma'$. The proof follows from Lemmas *2.5 (vii)* and *2.6*. ■

Notice that the definition of covering differs from that of lifting, in that $<\langle B_0,\tau_0\rangle,\langle B_1,\tau_1\rangle>$ is a lifting only if $B_1\leq_{\rm E} B_0$ and $\sigma_0\leq_{\rm E}\sigma_1$. For a covering, this is normally not the case.

Example 6.2.18 Since by 6.2.12 (i) $\quad\langle\{y{:}\varphi\},\varphi\rangle\prec\!\!\prec\langle\emptyset,\omega\rangle$,
by 6.2.12 (iv) also $\quad\langle\{x{:}\varphi\to\varphi',y{:}\varphi\},\varphi'\rangle\prec\!\!\prec\langle\{x{:}\omega\to\varphi'\},\varphi'\rangle$,

so by 6.2.12 (iii) $\langle\{x{:}\varphi\to\varphi'\},\varphi\to\varphi'\rangle \prec\!\!\prec \langle\{x{:}\omega\to\varphi'\},\omega\to\varphi'\rangle$,
and, again by 6.2.12 (iii), $\langle\emptyset,(\varphi\to\varphi')\to\varphi\to\varphi'\rangle \prec\!\!\prec \langle\emptyset,(\omega\to\varphi')\to\omega\to\varphi'\rangle$,
but not $(\varphi\to\varphi')\to\varphi\to\varphi' \leq_{\rm E} (\omega\to\varphi')\to\omega\to\varphi'$, *and neither* $(\omega\to\varphi')\to\omega\to\varphi' \leq_{\rm E} (\varphi\to\varphi')\to\varphi\to\varphi'$.

6.3 Completeness of operations

In this subsection, completeness of the above specified operations will be proved, both for the relevant as for the essential system. First the notion of chain of operations is introduced.

Definition 6.3.1 *i) A chain is an object* $[O_1,\ldots,O_n]$*, where each* O_i *is an operation of expansion, covering, substitution, or lifting, and*

$$[O_1,\ldots,O_n]\,(\langle B,\sigma\rangle) = O_n\,(\cdots(O_1\,(\langle B,\sigma\rangle))\cdots).$$

ii) On chains the operation of concatenation is denoted by $*$*, and*

$$[O_1,\ldots,O_i] * [O_{i+1},\ldots,O_n] = [O_1,\ldots,O_n].$$

iii) A relevant chain is a chain of expansions, concatenated with a chain consisting of at most one substitution, and at most one covering, in that order.

iv) An essential chain is a relevant chain, concatenated with one operation of lifting.

The next theorem shows that for every suitable pair for a term A, there exists a chain such that the result of the application of this chain to the principal pair of A produces the desired pair. Part *(i)* of the Lemmas *6.2.3*, and *6.2.10* are needed for the inductive step in case of an abstraction term, part *(iii.b)* of the proof, part *(ii)* of those lemmas are needed for the inductive step in case of an application term, part *(iii.c)*. Notice that, by construction, all operations mentioned in that part satisfy the conditions required by these lemmas.

Theorem 6.3.2 *If* $B \vdash_{\rm R} A{:}\sigma$ *and* $Pp\,(A) = \langle P,\pi\rangle$*, then there exists a relevant chain* C *such that* $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$.

Proof: By induction on the definition of $\vdash_{\rm R}$.

i) $\emptyset \vdash_{\rm R} A{:}\omega$. Take $Cov = \prec\langle P,\pi\rangle,\langle\emptyset,\omega\rangle\succ$, which, by Definition *6.2.13 (i)*, is a covering. Take

$$C = [Cov].$$

ii) $B \vdash_{\rm R} A{:}\sigma_1\cap\cdots\cap\sigma_n$. Then there are $B_1,\ldots,B_n$ such that for $1 \leq i \leq n$, $B_i \vdash_{\rm R} A{:}\sigma_i$, and $B = \Pi\{B_1,\ldots,B_n\}$. Let $Exp = \langle\pi,n,P,\pi\rangle$, then

$$Exp\,(\langle P,\pi\rangle) = \langle\Pi\{P_1,\ldots,P_n\},\pi_1\cap\cdots\cap\pi_n\rangle,$$

with $Pp\,(A) = \langle P_i,\pi_i\rangle$. By induction, there exist relevant chains $C_1,\ldots,C_n$ such that for $1 \leq i \leq n$, $C_i\,(\langle P_i,\pi_i\rangle) = \langle B_i,\tau_i\rangle$. Let, for $1 \leq i \leq n$, $C_i = \overrightarrow{Exp}_i * [Sub_i] * [Cov_i]$, and B'_i,τ'_i be such that

$$\overrightarrow{Exp}_i * [Sub_i]\,(\langle P_i,\pi_i\rangle) = \langle B'_i,\tau'_i\rangle, \text{ and } Cov_i = \prec\langle B'_i,\tau'_i\rangle,\langle B_i,\tau_i\rangle\succ.$$

Then, by Definition *6.2.12 (ii)*,

$$Cov = \prec\langle\Pi\{B'_1,\ldots,B'_n\},\tau'_1\cap\cdots\cap\tau'_n\rangle,\langle\Pi\{B_1,\ldots,B_n\},\tau_1\cap\cdots\cap\tau_n\rangle\succ$$

is a covering. Take

$$C = [Exp] * \overrightarrow{Exp}_1 * \cdots * \overrightarrow{Exp}_n * [Sub_1 \circ \cdots \circ Sub_n] * [Cov].$$

iii) $B \vdash_{\rm R} A{:}\sigma$, so $\sigma \in \mathcal{T}_{\rm s}$. This part is proved by induction on the structure of elements of $\mathcal{N}$. Notice that the case that $A \equiv \bot$ need not be considered.

a) $A \equiv x$. Then $B = \{x{:}\sigma\}$, $P = \{x{:}\varphi\}$, and $\pi = \varphi$. Take

$$C = [(\varphi \mapsto \sigma)].$$

b) $A \equiv \lambda x.A'$.

1) $x \in FV(A')$. Then there are α, β such that $\sigma = \alpha \to \beta$, $B, x{:}\alpha \vdash_{\mathrm{R}} A'{:}\beta$, and $Pp\,(\lambda x.A') = \langle P, \mu \to \pi \rangle$, where $Pp\,(A') = \langle P \cup \{x{:}\mu\}, \pi \rangle$. By induction there exists a relevant chain $C' = \overrightarrow{Exp} * [Sub] * [Cov']$ such that

$$C'\,(\langle P \cup \{x{:}\mu\}, \pi \rangle) = \langle B \cup \{x{:}\alpha\}, \beta \rangle.$$

Let α', β', B' be such that

$$\overrightarrow{Exp} * [Sub]\,(\langle P \cup \{x{:}\mu\}, \pi \rangle) = \langle B' \cup \{x{:}\alpha'\}, \beta' \rangle,$$

and

$$Cov' = \prec \langle B' \cup \{x{:}\alpha'\}, \beta' \rangle, \langle B \cup \{x{:}\alpha\}, \beta \rangle \succ.$$

Since $\beta \in \mathcal{T}_{\mathrm{s}}$, by construction also $\beta' \in \mathcal{T}_{\mathrm{s}}$ and, by Lemmas *6.2.10 (i)* and *6.2.3 (i)*, $\overrightarrow{Exp} * [Sub]\,(\langle P, \mu \to \pi \rangle) = \langle B', \alpha' \to \beta' \rangle$. Take

$$Cov = \prec \langle B', \alpha' \to \beta' \rangle, \langle B, \alpha \to \beta \rangle \succ,$$

which, by Definition *6.2.12 (iii)*, is a covering. Take

$$C = \overrightarrow{Exp} * [Sub] * [Cov].$$

2) $x \notin FV(A')$. Then there is β such that $\sigma = \omega \to \beta$, and $B \vdash_{\mathrm{R}} A'{:}\beta$. Then $Pp\,(\lambda x.A') = \langle P, \omega \to \pi \rangle$, where $Pp\,(A') = \langle P, \pi \rangle$. By induction there exists a relevant chain $C' = \overrightarrow{Exp} * [Sub] * [Cov']$ such that

$$C'\,(\langle P, \pi \rangle) = \langle B, \beta \rangle.$$

Let B', β' be such that

$$\overrightarrow{Exp} * [Sub]\,(\langle P, \pi \rangle) = \langle B', \beta' \rangle, \text{ and } Cov' = \prec \langle B', \beta' \rangle, \langle B, \beta \rangle \succ.$$

Since $\beta \in \mathcal{T}_{\mathrm{s}}$, by construction also $\beta' \in \mathcal{T}_{\mathrm{s}}$ and, by Lemma *6.2.10 (i)*, and *6.2.3 (i)*, $\overrightarrow{Exp} * [Sub]\,(\langle P, \omega \to \pi \rangle) = \langle B', \omega \to \beta' \rangle$. Take

$$Cov = \prec \langle B', \omega \to \beta' \rangle, \langle B, \omega \to \beta \rangle \succ,$$

which, by Definition *6.2.12 (iii)*, is a covering. Take

$$C = \overrightarrow{Exp} * [Sub] * [Cov].$$

c) $A \equiv x A_1 \cdots A_m$. Then there are $\sigma_1, \ldots, \sigma_m$, $B_1, \ldots, B_m$, such that, for every $1 \leq j \leq m$, $B_j \vdash_{\mathrm{R}} A_j{:}\sigma_j$, $B = \Pi\{B_1, \ldots, B_m, \{x{:}\sigma_1 \to \cdots \to \sigma_m \to \sigma\}\}$, and

$$P = \Pi\{P_1, \ldots, P_m, \{x{:}\pi_1 \to \cdots \to \pi_m \to \varphi\}\},\ \pi = \varphi,$$

with for every $1 \leq j \leq m$, $Pp\,(A_j) = \langle P_j, \pi_j \rangle$, in which φ does not occur. By induction, there are relevant chains $C_1, \ldots, C_m$ such that, for $1 \leq j \leq m$, $C_j\,(\langle P_j, \pi_j \rangle) = \langle B_j, \sigma_j \rangle$. Let, for $1 \leq j \leq m$, $C_j = \overrightarrow{Exp}_j * [Sub_j] * [Cov_j]$, and B'_j, σ'_j be such that

$$\overrightarrow{Exp}_j * [Sub_j]\,(\langle P_j, \pi_j \rangle) = \langle B'_j, \sigma'_j \rangle, \text{ and } Cov_j = \prec \langle B'_j, \sigma'_j \rangle, \langle B_j, \sigma_j \rangle \succ.$$

Let $\overrightarrow{Exp} = \overrightarrow{Exp}_1 * \cdots * \overrightarrow{Exp}_m$, and $Sub = Sub_1 \circ \cdots \circ Sub_m \circ (\varphi \mapsto \sigma)$, then, because of Lemmas *6.2.10 (ii)* and *6.2.3 (ii)*:

$$\begin{aligned}\overrightarrow{Exp} * [Sub]\,(\langle \Pi\{P_1, \ldots, P_m, \{x{:}\pi_1 \to \cdots \to \pi_m \to \varphi\}\}, \varphi \rangle) = \\ \langle \Pi\{B'_1, \ldots, B'_m, \{x{:}\sigma'_1 \to \cdots \to \sigma'_m \to \sigma\}\}, \sigma \rangle.\end{aligned}$$

Then, by Definition *6.2.12 (iv)*,

$$\begin{aligned}Cov = \prec \langle \Pi\{B'_1, \ldots, B'_m, \{x{:}\sigma'_1 \to \cdots \to \sigma'_m \to \sigma\}\}, \sigma \rangle \succ, \\ \langle \Pi\{B_1, \ldots, B_m, \{x{:}\sigma_1 \to \cdots \to \sigma_m \to \sigma\}\}, \sigma \rangle,\end{aligned}$$

is a covering. Take

$$C = \overrightarrow{Exp} * [Sub] * [Cov]. \qquad \blacksquare$$

Now, for essential type assignment the completeness of the here specified operations becomes easy to prove:

Theorem 6.3.3 If $B \vdash_{\rm E} A{:}\sigma$ and $Pp\,(A) = \langle P,\pi\rangle$, then there exists an essential chain C such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$.

Proof: By Lemma *6.1.2* there are B',σ' such that $B' \vdash_{\rm R} A{:}\sigma$, $\sigma' \leq_{\rm E} \sigma$, and $B \leq_{\rm E} B'$. By Theorem *6.3.2,* there exists a relevant chain C such that such that $C\,(\langle P,\pi\rangle) = \langle B',\sigma'\rangle$. Since $<\langle B',\sigma'\rangle,\langle B,\sigma\rangle>$ is a lifting, by Definition *6.3.1 (iv),* there exists an essential chain such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$. $\blacksquare$

Like in [9, 34, 4], it can be proved that there exists a precise relation between terms in $\mathcal{N}$ and principal pairs, both equipped with an appropriate ordering. This relation is in [34] defined using substitution of type-variables by the type constant ω. Using the notion of substitution defined here, this approach cannot be taken; instead, the relation $\prec\!\!\prec$ on pairs as given in Definition *6.2.12* is used.

Theorem 6.3.4 $\langle \mathcal{P}, \succ\!\!\succ\rangle$ is a meet semilattice isomorphic to $\langle \mathcal{N}, \sqsubseteq\rangle$.

Proof: Pp is, as function from $\mathcal{N}$ to $\mathcal{P}$, by Definition *6.1.5 (ii),* surjective. It is injective because of Theorems *4.2.4, 4.2.6,* and *6.3.3*. That Pp respects the order, i.e. if $A \sqsubseteq A'$, then $Pp\,(A') \prec\!\!\prec Pp\,(A)$, follows by straightforward induction. $\blacksquare$

Definition 6.3.5 (cf. [34, 4]) *i) Let M be a term. Let $\Pi(M)$ be the set of all principal pairs for all approximants of M: $\Pi(M) = \{Pp\,(A) \mid A \in \mathcal{A}(M)\}$.*

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 $\Pi(M)$.

The proof of the principal type property for the essential system is completed by the following:

Theorem 6.3.6 Let B and σ be such that $B \vdash_{\rm E} M{:}\sigma$.

i) $\mathcal{A}(M)$ is finite. Let $\langle P,\pi\rangle$ be the principal pair of M. Then there exists an essential chain C such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$.

ii) $\mathcal{A}(M)$ is infinite. Then there exist a pair $\langle P,\pi\rangle \in \Pi(M)$ and an essential chain C such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$. $\blacksquare$

Proof: From $B \vdash_{\rm E} M{:}\sigma$ and Theorem *3.2.7* follows $\exists\, A \in \mathcal{A}(M)\ [B \vdash_{\rm E} A{:}\sigma]$. Then:

i) By Definition *6.3.5,* there exists $A_M \in \mathcal{A}(M)$ such that $Pp\,(M) = Pp\,(A_M) = \langle P,\pi\rangle$. Since $Pp\,(A_M)$ is minimal in $\mathcal{P}$, so $Pp\,(A_M) \prec\!\!\prec Pp\,(A)$, by Theorem *6.3.4,* A_M is maximal in $\mathcal{A}(M)$, so $A \sqsubseteq A_M$. Then, by Lemma *2.5 (viii),* also $B \vdash_{\rm E} A_M{:}\sigma$.

ii) By Definition *6.3.5,* $\langle P,\pi\rangle = Pp\,(A) \in \Pi(M)$.

In any case, by Theorem *6.3.3,* there exists an essential chain such that $C\,(\langle P,\pi\rangle) = \langle B,\sigma\rangle$.

The same result, using relevant chains rather than essential chains, can be formulated for the relevant system.

Conclusions

This paper presented the essential intersection type assignment system, as a true restriction of the BCD-system that satisfies all properties of that system, where derivations are syntax-directed. Since all properties of the BCD-system are shown to hold for the essential system, the treatment of intersection types as in [6] has been too general. Instead of introducing $\cap$ as a general type-constructor, and ω as a type-constant, it is better to treat ω as the empty intersection, and to allow intersections only on the left of the $\rightarrow$.

References

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

[2] S. van Bakel. Partial Intersection Type Assignment of Rank 2 in Applicative Term Rewriting Systems. Technical Report 92-03, Department of Computer Science, University of Nijmegen, 1992.

[3] S. van Bakel. Essential Intersection Type Assignment. In R.K. Shyamasunda, editor, *Proceedings of FST&TCS '93. 13*$^{\text{th}}$ *Conference on Foundations of Software Technology and Theoretical Computer Science*, Bombay, India, volume 761 of *Lecture Notes in Computer Science*, pages 13–23. Springer-Verlag, 1993.

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

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

[6] 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.

[7] 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.

[8] M. Coppo, M. Dezani-Ciancaglini, F. Honsell, and G. Longo. Extended type structures and filter lambda models. In G. Lolli, G. Longo, and A. Marcja, editors, *Logic Colloquium 82*, pages 241–262, Amsterdam, the Netherlands, 1984. North-Holland.

[9] 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.

[10] 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.

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

[12] M. Coppo and P. Giannini. A complete type inference algorithm for simple intersection types. In J.-C. Raoult, editor, *Proceedings of CAAP '92. 17th Colloqim on Trees in Algebra and Programming*, Rennes, France, volume 581 of *Lecture Notes in Computer Science*, pages 102–123. Springer-Verlag, 1992.

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

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

[15] F. Damiani and P. Giannini. A Decidable Intersection Type System based on Relevance. In M. Hagiya and J.C. Mitchell, editors, *Proceedings of TACS '94. International Symposium on Theoretical Aspects of Computer Software*, Sendai, Japan, volume 789 of *Lecture Notes in Computer Science*, pages 707–725. Springer-Verlag, 1994.

[16] M. Dezani-Ciancaglini and I. Margaria. F-semantics for intersection type discipline. In G. R. Kahn, D. B. Macqueen, and G. Plotkin, editors, *Semantics of data types. International symposium*, Sophia-Antipolis, France, volume 173 of *Lecture Notes in Computer Science*, pages 279–300. Springer-Verlag, 1984.

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

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

[19] P. Gardner. Discovering Needed Reductions Using Type Theory. In M. Hagiya and J.C. Mitchell, editors, *Proceedings of TACS '94. International Symposium on Theoretical Aspects of Computer Software,* Sendai, Japan, volume 789 of *Lecture Notes in Computer Science*, pages 555–597. Springer-Verlag, 1994.

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

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

[22] 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.

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

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

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

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

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

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

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] W.W. Tait. Intensional interpretation of functional of finite types. *Journal of Symbolic Logic*, 32, 1967.

[37] 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.