Strict Intersection Types for the Lambda Calculus

This version has all proofs included, rather than accumulated in an appendix.

(ACM Computing Surveys, 43(3) article 20, April 2011)

Steffen van Bakel

Department of Computing, Imperial College London,
180 Queen's Gate, London SW7 2BZ, U.K.
svb@doc.ic.ac.uk

Contents

1 Introduction

In the recent years several notions of type assignment for several (extended) lambda calculi have been studied. The oldest among these is known as the Curry type assignment system [26] (see also [39]). Formulated via natural deduction[1], using the arrow type constructor '$\rightarrow$', it expresses abstraction and application, and can be used to obtain a (basic) functional characterisation of terms. It is well known that in Curry's system, the problem of typeability

Given a term M, are there a context Γ and a type σ such that $\Gamma \vdash M:\sigma$?

is decidable, and for this reason it gained popularity as the basis for modern type systems for (functional) programming languages, like Haskell [41] and ML [48]. Although Curry's system is already powerful, in that many programs can be typed with it[2], it has drawbacks. It is, for example, not possible to assign a type to the term $\lambda x.xx$, and some β-equal terms can be assigned different types - an example will be given at the end of Section 3.1.

The Intersection Type Discipline is an extension of Curry's system that does not have these drawbacks. The extension made consists mainly of allowing for term-variables (and terms) to have more than one type, joined via the intersection type constructor '$\cap$, that gets added next to the type constructor '$\rightarrow$' of Curry's system, as well as the type constant 'ω', which is the universal type and is used for otherwise untypeable terms or terms that disappear during reduction. This simple extension made the proof of many strong semantic and characterisation results achievable for the λ-calculus, the most important of which we will discuss here in the context of strict intersection types.

This survey starts in Section 3, where we will discuss a number of papers that treat intersection types, focussing on their main results. We first look at the system defined by [20] that introduced intersection types[3] ('$\vdash_{\rm CD}$', see Definition 3.2). From this initial system several others emerged; the best known and most frequently quoted is the BCD-system ('$\vdash_{\rm BCD}$', see Definition 3.10) as presented by Barendregt, Coppo, and Dezani-Ciancaglini in [15], but

[1] Curry's system ('$\vdash_{\rm C}$', see Definition 3.1) enjoys the Curry-Howard correspondence – also attributed to de Bruijn – with respect to implicative intuitionistic logic.

[2] This problem is normally addressed via more expressive type systems, using polymorphism and recursion [27], but these can be mapped back onto the Curry system, using a translation of programs into λ-terms, and adding the black box fixed point combinator $\mathbf{Y}$ with type $(A\rightarrow A)\rightarrow A$, for all A.

[3] That paper uses the word 'sequence' instead of 'intersection'.

there are two earlier papers [22, 21] that investigate interesting systems that can be regarded as in-between '$\vdash_{\text{CD}}$' and '$\vdash_{\text{BCD}}$'. In the first, Coppo, Dezani-Ciancaglini, and Venneri present two type assignment systems, that, in approach, are more general than '$\vdash_{\text{CD}}$': in addition to the type constructors '$\rightarrow$' and '$\cap$', they also contain the type constant 'ω'[4]. The first system presented in [22] ('$\vdash_{\text{CDV}}$', see Definition 3.3) is a true extension of '$\vdash_{\text{CD}}$'; the second one ('$\vdash_{\text{R}}$', see Definition 3.6) limits the use of intersection types in contexts.

The BCD-system is based on '$\vdash_{\text{CDV}}$'; it generalises intersection types 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. To treat extensionality, it also introduces a type inclusion relation '$\leq$' that is contra-variant in arrow types, and closes type assignment for this relation.

As mentioned above, the slight generalisation of allowing for terms to have more than one type causes a great change in complexity; in fact, now all terms having a (head-)normal form can be characterised by their assignable types (see Section 6.3), a property that immediately shows that type assignment (even in the system that does not contain ω, see Section 8.1) is undecidable. Also, by introducing this extension a system is obtained that is closed under β-equality: if $\Gamma \vdash M{:}\sigma$ and $M =_\beta N$, then $\Gamma \vdash N{:}\sigma$ (see Section 4.2). This property is exploited in the main contribution of [15] to the theory of intersection types when it shows that intersection types can be used for semantics: it introduces a filter λ-model and shows completeness of type assignment.

Another way to define semantics for the λ-calculus is through approximation: as in [59, 14], the set of terms can be extended by adding the term-constant $\bot$, which leads to the notion of *approximate normal forms* that are in essence finite rooted segments of Böhm-trees [14]. It is well known that interpreting a term by its Böhm tree gives a model for the Lambda Calculus, and so the same is possible when using the set of its approximants. From the Approximation Theorem, i.e. the observation that there exists a very precise relation between types assignable to a term M and those assignable to its approximants, $\mathcal{A}(M)$, formulated as

$$\Gamma \vdash M{:}\sigma \iff \exists\, A \in \mathcal{A}(M)\ [\Gamma \vdash A{:}\sigma]$$

(see [21, 55, 3, 7] and Section 6.1 and 9.4), it is immediately clear that the set of intersection types assignable to a term can be used to define a model for the Lambda Calculus (see [15, 3, 7] and Section 7.1).

Another reason for the popularity of Curry's system within the context of programming languages is that it has the principal pair[5] property:

If M is typeable, then there are Π,π such that $\Pi \vdash M{:}\pi$, and, for every Γ,σ such that $\Gamma \vdash M{:}\sigma$, there exist a way of (constructively) generating $\langle \Gamma,\sigma\rangle$ from $\langle \Pi,\pi\rangle$.

Historically, principal types were first studied by [36] in the context of Combinator Systems, exploiting successfully for the first time [53]'s notion of unification in the context of type theory. This property found its way into programming, mainly through the pioneering work of [48] (see also [27]). He introduced a functional programming language ML, of which the underlying type system is an extension of Curry's system, using Hindley's approach[6]. The extension consists of the introduction of polymorphic functions, i.e. functions that can be

[4] The type constant ω was first introduced by [56]; a joint paper appeared as [19].

[5] In the past, often the terminology 'principal type' was used, but 'principal pair' expresses this property better; see also [60] and [44] for discussions on principal types, pairs and typings.

[6] This type system is sometimes also called the Hindley-Milner system.

applied to various kinds of arguments, even of incomparable type. The formal motivation of this concept lies directly in the notion of principal types [17].

A disadvantage of '$\vdash_{\text{BCD}}$' (and of any real intersection system, for that matter) is that type assignment is undecidable and it therefore cannot be used directly for programming languages. For this reason, in recent years, some decidable restrictions have been studied. The first was the Rank2 intersection type assignment system [5], as first suggested by [47], that is very close to the notion of type assignment as used in ML. The key idea for this system is to restrict the set of types to those of the shape $(\sigma_1 \cap \cdots \cap \sigma_n) \rightarrow \tau$, where the σ_i are types that do not contain intersections. This kind of type later was used outside the context of the λ-calculus [8, 9, 28, 30, 58]; many decidable restrictions of various ranks were later defined by [45], [46].

That intersection types can be used as a basis for programming languages was first discussed by [51]. This led to the development of the (typed, i.e. terms contain types syntactically) programming language Forsythe [52], and to the work of Pierce [1991, 1993], 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. Intersection types have found their use also in the context of abstract interpretation [42, 43, 24, 16].

Another disadvantage of '$\vdash_{\text{BCD}}$' is that it is too general: 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 [15] 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 '$\sim$'. This makes the problem of inhabitation (is there a closed term that has this type).

Building on the definition of principal context and type scheme introduced in [21], [55] showed that '$\vdash_{\text{BCD}}$' has the principal pair property. But, although, for every M, the set $\{ \langle \Gamma, \sigma \rangle \mid \Gamma \vdash M : \sigma \}$ can be generated using operations specified in that paper, the problem of type-checking

Given a term M and type σ, is there a context Γ such that $\Gamma \vdash_{\text{BCD}} M : \sigma$?

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

The strict type assignment system ('$\vdash_{\text{S}}$', see Definition 5.1) as defined in [3] (a first version appeared in [2], that coined the moniker *strict*) is a restriction of '$\vdash_{\text{BCD}}$'; it uses a set of strict types, a variant of intersection types that has been used in many papers since, and that are actually the normalised tail-proper types of [22]. Although there are rather strong restrictions imposed, the provable results for '$\vdash_{\text{S}}$' are very close to those for '$\vdash_{\text{BCD}}$'. For example, the sets of normalisable terms and those having a normal form can be equally elegantly characterised. The main difference between the two systems is that '$\vdash_{\text{S}}$' is *not* extensional (closed for η-reduction), whereas '$\vdash_{\text{BCD}}$' is.

'$\vdash_{\text{S}}$' gives rise to a strict filter λ-model that satisfies all major properties of the filter λ-model as presented in [15], but is an essentially different λ-model, equivalent to [33]'s model $\mathcal{D}_A$. With the use of the inference type semantics, in [3] soundness and completeness for strict type assignment was proven, without having the necessity of introducing '$\leq$' (see also Section 5).

This paper will show the usefulness and elegance of strict intersection types. We will focus on the notion of *essential* intersection type assignment ('$\vdash_{\text{E}}$', see Definition 4.3) for the Lambda Calculus that was first defined in [5, 7] (albeit in different notation), and later studied in [11];

we will revisit the results of those papers, give in part new proofs, and briefly compare it with other existing systems. '$\vdash_{\rm E}$' is a true restriction of '$\vdash_{\rm BCD}$' that satisfies all properties of that system, and is an extension of '$\vdash_{\rm CD}$'. It is also an extension of '$\vdash_{\rm S}$'; the major difference is that it, like '$\vdash_{\rm BCD}$', will prove to be closed for η-reduction: if $\Gamma \vdash_{\rm E} M : \sigma$ and $M \to_{\eta} N$, then $\Gamma \vdash_{\rm E} N : \sigma$.

By establishing all major properties in the context of strict types, we will show that the treatment of intersection types as in [15] has been too general; the same results can be 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 the 'strict' approach are a less complicated type language, less complicated proofs, and more precise and elegant definitions. For example, the operations that are defined [7] (see Section 10), needed in a proof for the principal pair property, in particular that of expansion, are less complicated; moreover, they are 'orthogonal': they do not overlap. In addition, in order to prove a completeness result using intersection types, there is no need to be as general as in [15]; this result can also be obtained for '$\vdash_{\rm E}$' (see Section 7.2).

In previous papers, the Approximation Theorem and Strong Normalisation Theorem were proven independently [7, 3], though both using [57]'s technique of Computability Predicates. This technique has been widely used to study normalisation properties or similar results, as for example in [23, 31, 54]. In this survey, we will show that both are special cases of a more fundamental result, using a variant of the technique developed in [12] for Term Rewriting, that has also found its application in the field of Combinator Systems in [13]. This more fundamental result, first published in [11], consists of defining a notion of reduction on derivations that generalises cut-elimination, and the proof of the theorem that this kind of reduction is strongly normalisable; a similar result for '$\vdash_{\rm S}$' was published in [10]. For intersection systems, there is a significant difference between derivation reduction and ordinary reduction (see Section 9.2); unlike normal typed or type assignment system, in '$\vdash_{\rm E}$' not every term redex occurs with types in a derivation. Moreover, especially the use of a contra-variant relation '$\leq$' on types, together with an associated derivation rule, greatly disturbs the smoothness of proofs (see again Section 9.2).

From this strong normalisation result for derivation reduction, the Approximation Theorem and Strong Normalisation Theorem follow easily. The first of these implies the Head-Normalisation Theorem and (indirectly) the Normalisation Theorem, as was already shown in [7] (see Section 6).

Some results already known to hold for, for example, '$\vdash_{\rm BCD}$', will be shown to hold for '$\vdash_{\rm E}$':

- If $\Gamma \vdash_{\rm E} M : \sigma$ and $M =_{\beta} N$, then $\Gamma \vdash_{\rm E} N : \sigma$.
- If $\Gamma \vdash_{\rm E} M : \sigma$ and $M \to_{\eta} N$, then $\Gamma \vdash_{\rm E} N : \sigma$.
- $\Gamma \vdash_{\rm E} M : \sigma$ and $\sigma \neq \omega$, if and only if M has a head-normal form.
- $\Gamma \vdash_{\rm E} M : \sigma$ and ω does not occur in Γ and σ, if and only if M has a normal form.
- $\Gamma \vdash_{\rm E} M : \sigma$ and ω is not used at all, if and only if M is strongly normalisable.
- $\Gamma \vdash_{\rm E} M : \sigma$ if and only it there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\rm E} A : \sigma$.
- '$\vdash_{\rm E}$' has the principal pair property.
- cut-elimination is strongly normalisable, and the characterisation properties are all consequences of this result.

These properties and their proofs will be reviewed here.

Paper outline

In Section 2, we repeat some definitions for the λ-calculus that are relevant to this paper, and introduce the notion of approximation and approximants. In Section 4, we will recall the definition of the essential type assignment system of [7], together with some of its main properties. This system is placed in the context of other, previously published intersection systems in Section 3, like the initial Coppo-Dezani system, and the well-known BCD-system, and the strict system in Section 5.

In Section 6 we will prove the approximation result, and show that it leads to the characterisation of head-normalisability and normalisability. The relation between the essential and the BCD-system is shown in Section 7, followed by the proof of completeness of type assignment. This will be followed in Section 8 by a new proof for the property that, in the system without ω, the typeable and strongly normalisable terms coincide.

In Section 9.2, a notion of reduction on derivations in the essential system is defined, for which we will show a strong normalisation result in Section 9.3. This result will lead in Section 9.4 to new proofs for the approximation and strong-normalisation result; we will briefly discuss the proof for the strong normalisation of derivation reduction in the strict and in the relevant system. Finally, in Section 10, we will discuss the proof for the principal pair property for the essential system.

The larger part of the contents of this survey has appeared in [2, 3, 5, 6, 4, 7, 10, 11].

Notations

In this survey, the symbol φ will be a type-variable; Greek symbols like α, β, ϕ, ψ, ρ, σ, and τ will range over types, and π will be used for principal types. The type constructor '$\rightarrow$' will be assumed to associate to the right, and '$\cap$' binds stronger than '$\rightarrow$'. M, N, P, Q are used for λ-terms; x, y, z for term-variables; $M[N/x]$ for the usual operation of substitution on λ-terms; A for terms in $\lambda\bot$-normal form. Γ is used for contexts, $\Gamma\backslash x$ for the context obtained from Γ by erasing the statement that has x as subject, and Π for principal contexts. Two types (contexts, pairs of context and type) are *disjoint* if and only if they have no type-variables in common. All symbols can appear indexed.

Notions of type assignment are defined as ternary relations on contexts, terms, and types, that are denoted by '$\vdash$', possibly indexed if necessary. If in a notion of type assignment for M there are context Γ and type σ such that $\Gamma \vdash M{:}\sigma$, then M is *typed with* σ, and σ is *assigned to* M. We will write $\underline{n}$ for the set $\{1,\ldots,n\}$, and will often use a vector notation '$\overrightarrow{\cdot}$' for the purpose of abbreviation. For example, $P\overrightarrow{M_i}$ stands for $PM_1\cdots M_n$ for a suitable n, and $[N_1/x_1,\ldots,N_n/x_n]$ is abbreviated by $[\overrightarrow{N_i/x_i}]$.

2 The λ-calculus

We assume the reader to be familiar with the λ-calculus [18, 14]; we just recall the definition of λ-terms and β-contraction and some notions that are relevant to this survey.

Definition 2.1 (λ-TERMS, FREE AND BOUND VARIABLES, AND SUBSTITUTION)

i) The set Λ of *λ-terms* is defined by the grammar:

$$M,N ::= x \mid \lambda x.M \mid MN$$

A (term)-context C[ ] is a term with a unique hole, obtained by removing a subterm.

ii) The set of *free* variables and of *bound* variables are defined by:

$$\begin{array}{llll} fv(x) & = \{x\} & bv(x) & = \emptyset \\ fv(M_1M_2) & = fv(M_1)\bigcup fv(M_2) & bv(M_1M_2) & = bv(M_1)\bigcup bv(M_2) \\ fv(\lambda y.M) & = fv(M)\backslash\{y\} & bv(\lambda y.M) & = bv(M)\bigcup\{y\} \end{array}$$

iii) The replacement of the free occurrences of x in M by N, $M[N/x]$, is defined by:

$$\begin{array}{rclcrcl} x[N/x] &=& N & \qquad & (M_1M_2)[N/x] &=& M_1[N/x]M_2[N/x] \\ y[N/x] &=& y, \text{ if } y \neq x & & (\lambda y.M)[N/x] &=& \lambda y.(M[N/x]) \end{array}$$

This notion is normally assumed to be *capture avoiding*, i.e. in the last case, y does not occur free in N.

Definition 2.2 (β-CONTRACTION) *i*) The reduction relation $\rightarrow_\beta$ is defined as the contextual (i.e. compatible [14]) closure of the rule:

$$(\lambda x.M)N \rightarrow_\beta M[N/x]$$

ii) The reduction relation $\twoheadrightarrow_\beta$ is defined as the reflexive, transitive closure of $\rightarrow_\beta$, and $=_\beta$ as the equivalence relation generated by $\twoheadrightarrow_\beta$.

iii) The λI-calculus is defined by restricting binding to: if $\lambda x.M \in \Lambda I$, then $x \in fv(M)$.

To guarantee that reduction is capture-free, α-conversion (renaming of bound variables) is to take place silently.

Normal forms and head-normal forms are defined as follows:

Definition 2.3 *i*) The set $\mathcal{N} \subset \Lambda$ of *normal forms* that is defined by the grammar:

$$N ::= xN_1 \cdots N_n \ (n \geq 0) \mid \lambda x.N$$

ii) The set $\mathcal{H}$ of *head-normal forms* is defined by:

$$H ::= xM_1 \cdots M_n \ (n \geq 0) \mid \lambda x.H$$

where the M_i $(i \in \underline{n})$ are arbitrary λ-terms.

We will use the following notions:

Definition 2.4 A term M is called:

i) *normalisable* if there exists an $N \in \mathcal{N}$ such that $M \twoheadrightarrow_\beta N$.

ii) *head-normalisable* if there exists an $H \in \mathcal{H}$ such that $M \twoheadrightarrow_\beta H$.

iii) *strongly normalisable* if all reduction paths starting from M are finite.

2.1 Approximate normal forms

The notion of approximant for λ-terms was first presented by [59], and is defined using the notion of terms in $\lambda\bot$-normal form[7].

Definition 2.5 *i*) The set $\Lambda\bot$ of *$\lambda\bot$ -terms* is defined by:

$$M ::= x \mid \bot \mid \lambda x.M \mid M_1M_2$$

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

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

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

$$A ::= \bot \mid \lambda x.A \ (A \neq \bot) \mid xA_1 \cdots A_n \ (n \geq 0)$$

[7] Like in [14], we use $\bot$ (called *bottom*) instead of Ω; also, the symbol $\sqsubseteq$ is used for a relation on $\lambda\bot$-terms, inspired by a similar relation defined on Böhm-trees in [14].

Definition 2.6 (Approximants) *i*) The partial order $\sqsubseteq \subseteq (\Lambda\bot)^2$ is defined as the smallest pre-order (i.e. reflexive and transitive relation) such that:

$$\begin{array}{rcl} \bot & \sqsubseteq & M \\ M \sqsubseteq M' & \Rightarrow & \lambda x.M \sqsubseteq \lambda x.M' \\ M_1 \sqsubseteq M_1' \ \& \ M_2 \sqsubseteq M_2' & \Rightarrow & M_1 M_2 \sqsubseteq M_1' M_2' \end{array}$$

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

iii) The relation $\underset{\sim}{\sqsubset} \subseteq \mathcal{A} \times \Lambda\bot$ is defined by:

$$A \underset{\sim}{\sqsubset} M \iff \exists M' \, [M' =_\beta M \ \& \ A \sqsubseteq M']$$

If $A \underset{\sim}{\sqsubset} M$, then A is called an *approximant* of M.

iv) $\mathcal{A}(M) = \{A \in \mathcal{A} \mid A \underset{\sim}{\sqsubset} M\}$.

The following properties of approximants hold and are needed below:

Lemma 2.7 *i) If $A \in \mathcal{A}(xM_1 \cdots M_n)$, $A \neq \bot$ and $A' \in \mathcal{A}(N)$, then $AA' \in \mathcal{A}(xM_1 \cdots M_n N)$.*

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

- *$A \equiv A'z$, $z \notin fv(A')$, and $A' \in \mathcal{A}(M)$, or*
- *$\lambda z.A \in \mathcal{A}(M)$.*

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

Proof: Easy. ■

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

Definition 2.8 *i*) On $\Lambda\bot$, the partial mapping *join*, $\sqcup$: $\Lambda\bot \times \Lambda\bot \to \Lambda\bot$, is defined by:

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

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

The last alternative in the definition of '$\sqcup$' defines the join on applications in a more general way than that of [35], that would state that

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

since it is not always sure if a join of two arbitrary terms exists. However, that is only an issue if the function should be total, which is not the case here; our more general definition will only be used on terms that are compatible, so the conflict is only apparent.

Remark 2.9 Because of 2.7(*iii*), $\mathcal{A}(M)$ can be used to define a semantics for the Lambda Calculus. In fact, it is possible to show that

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

where $BT(M)$ stands for the *Böhm tree* of M, a tree that represents the (possible infinite) normal form of M (see [14]).

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

Lemma 2.10 *If $M_1 \sqsubseteq M$, and $M_2 \sqsubseteq M$, then $M_1 \sqcup M_2$ is defined, and:*

$$M_1 \sqsubseteq M_1 \sqcup M_2, M_2 \sqsubseteq M_1 \sqcup M_2, \textit{and } M_1 \sqcup M_2 \sqsubseteq M.$$

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

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

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

iii) If $M_1 \equiv P_1Q_1$, then $M \equiv PQ$, $P_1 \sqsubseteq P$, $Q_1 \sqsubseteq Q$, and either $M_2 = \bot$ or $M_2 \equiv P_2Q_2$. Again, the first case has been dealt with in part (*i*), and for the other: then $P_2 \sqsubseteq P$, $Q_2 \sqsubseteq Q$. By induction, we know $P_1 \sqsubseteq P_1 \sqcup P_2$, $P_2 \sqsubseteq P_1 \sqcup P_2$, and $P_1 \sqcup P_2 \sqsubseteq P$, as well as $Q_1 \sqsubseteq Q_1 \sqcup Q_2$, $Q_2 \sqsubseteq Q_1 \sqcup Q_2$, and $Q_1 \sqcup Q_2 \sqsubseteq Q$. Then we have also $P_1Q_1 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, $P_2Q_2 \sqsubseteq (P_1 \sqcup P_2)(Q_1 \sqcup Q_2)$, and $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \sqsubseteq PQ$. Notice that, by definition, $(P_1 \sqcup P_2)(Q_1 \sqcup Q_2) \equiv (P_1Q_1) \sqcup (P_2Q_2)$. ■

Notice that $M_1 \sqsubseteq (M_1 \sqcup M_2) \sqsubseteq (M_1 \sqcup M_2 \sqcup M_3) \sqsubseteq \cdots$, so it is natural to consider $\bot$ to be the empty join, i.e. if $M \equiv M_1 \sqcup \cdots \sqcup M_n$, and $n = 0$, then $M \equiv \bot$.

Lemma 2.11 *i) If $M \sqsubseteq M_i$, for all $i \in \underline{n}$, then $M \sqsubseteq M_1 \sqcup \cdots \sqcup M_n$.*

ii) If $M \sqsubseteq N$, and $N \sqsubseteq P$, then $M \sqsubseteq P$.

iii) If $M \sqsubseteq M_1M_2$, and $M \neq \bot$, then there are M_3, M_4 such that $M = M_3M_4$, and both $M_3 \sqsubseteq M_1$, $M_4 \sqsubseteq M_2$.

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

3 A historical perspective

In this section, we discuss briefly the various intersection systems as they appeared, in order to be able to compare them with '$\vdash_{\mathrm{E}}$' that we will present in Section 4.

3.1 Curry Type Assignment

Type assignment for the Lambda Calculus was first studied by [25] (see also [26, 39]). Curry's original system expresses abstraction and application, and types are built over the single type constant *o* using the type constructor '$\rightarrow$' (*arrow*). The system we will present in this section is a generalisation of that system in that types are built over type variables and terms have infinitely many types.

Definition 3.1 *i*) Let $\mathcal{V}$ be a countable (infinite) set of type-variables, ranged over by φ. The set of *Curry-types* is inductively defined by the following grammar:

$$\sigma,\tau ::= \varphi \mid (\sigma\rightarrow\tau)$$

ii) *Curry-type assignment* is defined by the following natural deduction system.

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

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

iii) A *context* is a partial mapping from term-variables to types, normally written as a set of statements of the shape $x{:}\sigma$, where the x are distinct; we write $\Gamma,x{:}\sigma$ for the context $\Gamma \cup \{x{:}\sigma\}$, when x does not occur in Γ.

iv) We write $\Gamma \vdash_{\mathrm{C}} M : \tau$ if there exists a derivation built using the above rules with conclusion $M : \tau$ and the context Γ contains at least all the not cancelled statements in that derivation (notice that, in rule $(\to I)$, the statement $x{:}\sigma$ is cancelled).

As used above (and as also used in [22, 20, 15, 3, 7]), the notation for the type assignment rules is that of *natural deduction*. A disadvantage of that notation is that the precise structure of contexts, i.e. the collection of statements for the free variables on which the typing of a term depends, is left unspecified; in intersection systems that use ω, not every term variable necessarily carries a type, and the content of the context is not as obvious. So, rather, in this survey we opt for a sequent-style notation (as also used in [10, 11]), always explicitly stating the context:

$$(\to I) : \frac{\Gamma, x{:}\sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau} \qquad (\to E) : \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

Notice that, in rule $(\to I)$, the cancellation of $x{:}\sigma$ is expressed by removing it from the context, since the conclusion no longer depends on it; we then also need a rule that deals with variables:

$$\frac{}{\Gamma, x{:}\sigma \vdash x : \sigma}$$

stating the extraction of a statement for a variable from the context.

The main results proven for this system are:

- The principal pair property: for every typeable M there is a pair $\langle \Pi, \pi \rangle$, such that: $\Pi \vdash_{\mathrm{C}} M : \pi$, and for every pair $\langle \Gamma, \sigma \rangle$ such that $\Gamma \vdash_{\mathrm{C}} M : \sigma$, there exists a type-substitution S (replacing type variables by types) such that $S(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.
- Decidability of type assignment: there exists an effective algorithm that, give a term M, will return a pair $\langle \Gamma, \sigma \rangle$, i.e. such that $\Gamma \vdash_{\mathrm{C}} M : \sigma$ if M is typeable - in fact, it then returns the principal pair.
- Strong normalisability of all typeable terms; this is also a result of Theorem 8.15.

Curry's system has drawbacks: it is for example not possible to assign a type to the λ-term $\lambda x.xx$, and although the λ-terms $\lambda cd.d$ and $(\lambda xyz.xz(yz))(\lambda ab.a)$ are β-equal, the principal type schemes for these terms are different, respectively $\varphi_0 \to \varphi_1 \to \varphi_1$ and $(\varphi_1 \to \varphi_0) \to \varphi_1 \to \varphi_1$. The Intersection Type Discipline as presented below 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; below, in order to develop a concise overview of the field, various systems will be shown.

3.2 The Coppo-Dezani type assignment system

Intersection type assignment is first introduced by [20]. The system presented in that paper is a true extension of Curry's system, by allowing for more than one type for term-variables in the $(\to I)$-derivation rule and therefore to also allow for more than one type for the right-hand term in the $(\to E)$-derivation rule.

Definition 3.2 (c.f. [20]) *i*) The set of types is inductively defined by:

$$\begin{array}{rcl}\phi,\psi &::=& \varphi \mid \sigma{\rightarrow}\psi \\ \sigma,\tau &::=& \phi_1{\cap}\cdots{\cap}\phi_n \;\; (n \geq 1)\end{array}$$

ii) *Type assignment* is defined by the following natural deduction system:

$$(\cap E):\ \frac{}{\Gamma,x{:}\phi_1{\cap}\cdots{\cap}\phi_n \vdash x{:}\phi_i}\ (i \in \underline{n}) \qquad (\cap I):\ \frac{\Gamma \vdash M{:}\phi_1 \quad \cdots \quad \Gamma \vdash M{:}\phi_n}{\Gamma \vdash M{:}\phi_1{\cap}\cdots{\cap}\phi_n}\ (n \geq 1)$$

$$(\rightarrow I):\ \frac{\Gamma,x{:}\sigma \vdash M{:}\phi}{\Gamma \vdash \lambda x.M{:}\sigma{\rightarrow}\phi} \qquad (\rightarrow E):\ \frac{\Gamma \vdash M{:}\sigma{\rightarrow}\phi \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash MN{:}\phi}$$

We write $\Gamma \vdash_{\text{CD}} M{:}\sigma$ for statements derivable in this system.

The main properties of that system proven in [20] are:

- subject reduction: If $\Gamma \vdash_{\text{CD}} M{:}\sigma$, and $M \rightarrow_\beta N$, then $\Gamma \vdash_{\text{CD}} N{:}\sigma$.
- normalisability of typeable terms: If $\Gamma \vdash_{\text{CD}} M{:}\sigma$, then M has a normal form.
- typeability of all terms in normal form.
- closure for β-equality in the λI-calculus: if $\Gamma \vdash_{\text{CD}} M{:}\sigma$, and $M =_{\beta I} N$, then $\Gamma \vdash_{\text{CD}} N{:}\sigma$.
- in the λI-calculus: $\Gamma \vdash_{\text{CD}} M{:}\sigma$ if and only if M has a normal form.

This system is not closed for β-expansion for the full λ-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 $\Gamma \vdash_{\text{CD}} N{:}\rho$, the redex $(\lambda x.M)N$ cannot be typed.

3.3 The Coppo-Dezani-Venneri type assignment systems

In [22] two type assignment systems are presented by Coppo, Dezani-Ciancaglini, and Venneri, that, in approach, are more general than '$\vdash_{\text{CD}}$': in addition to the type constructors '$\rightarrow$' and '$\cap$', they also contain the type constant 'ω', as first introduced by [56]. The first system presented in [22] is a true extension of the one presented in Definition 3.2. The second one is a relevant restriction.

Definition 3.3 (C.F. [22]) *i*) The set of types is inductively defined by:

$$\begin{array}{rcl}\phi,\psi &::=& \varphi \mid \omega \mid \sigma{\rightarrow}\psi \\ \sigma,\tau &::=& \phi_1{\cap}\cdots{\cap}\phi_n \quad (n \geq 1)\end{array}$$

ii) Every non-intersection type is of the shape $\sigma_1{\rightarrow}\cdots{\rightarrow}\sigma_n{\rightarrow}\phi$, where $\sigma_1,\ldots,\sigma_n$ are intersections, and ϕ is a type-variable or ω. The type is called *tail-proper* if $\phi \neq \omega$.

iii) *Type assignment* is defined by:

$$(\cap E):\ \frac{}{\Gamma,x{:}\phi_1{\cap}\cdots{\cap}\phi_n \vdash x{:}\phi_i}\ (i \in \underline{n}) \qquad (\omega):\ \frac{}{\Gamma \vdash M{:}\omega} \qquad (\rightarrow I):\ \frac{\Gamma,x{:}\sigma \vdash M{:}\psi}{\Gamma \vdash \lambda x.M{:}\sigma{\rightarrow}\psi}$$

$$(\cap I):\ \frac{\Gamma \vdash M{:}\phi_1 \quad \cdots \quad \Gamma \vdash M{:}\phi_n}{\Gamma \vdash M{:}\phi_1{\cap}\cdots{\cap}\phi_n}\ (n \geq 1) \qquad (\rightarrow E):\ \frac{\Gamma \vdash M{:}\sigma{\rightarrow}\phi \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash MN{:}\phi}$$

We write $\Gamma \vdash_{\text{CDV}} M{:}\sigma$ for statements derivable in this system.

As remarked above, the original presentation of this system used natural deduction rules, where the context used is implicit; as a result, rule $(\cap E)$ was not part of that definition, but is added here since we have switched to a sequent style presentation. Also, the original presentation for rule $(\rightarrow I)$

$$(\to I): \quad \dfrac{\begin{array}{ccc}[x{:}\sigma_1] & \cdots & [x{:}\sigma_n]\\ & \vdots & \\ & M:\tau & \end{array}}{\lambda x.M:\rho\to\tau}\ (\rho \textit{ is a sequence at least containing } \sigma_1,\ldots,\sigma_n)$$

allowed for a number of statements for a bound variable to be combined (including some that are not used in the derivation), so has rule $(\cap E)$ implicitly present.

The changes with respect to '$\vdash_{\text{CD}}$' are small, but important. By introducing the type constant ω next to the intersection types, a system is obtained that is closed under β-equality for the *full* λ-calculus; ω is used to type those sub-terms that disappear during reduction, and that cannot be typed from the current context. This addition also makes all terms having a *head-normal form* typeable.

Recognising an undesired complexity in the language of types, that paper also introduces a notion of normalisation for types.

Definition 3.4 (Normalised types) The set of *normalised types* is inductively defined in [22] by:

i) All type-variables are normalised types.
ii) If σ is a normalised intersection type and ψ is a normalised type, then $\sigma\to\psi$ is a normalised type.
iii) ω is a normalised intersection type.
iv) If $\phi_1,\ldots,\phi_n$ are normalised types $(n \geq 1)$, then $\phi_1\cap\cdots\cap\phi_n$ is a normalised intersection type.

Observe that the only normalised non-tail-proper type is ω, and that, if $\sigma\to\psi$ is a normalised type, then so is ψ, so, in particular, $\psi \neq \omega$.

The main properties of this system proven in [22] are:

- If $\Gamma \vdash_{\text{CDV}} M{:}\sigma$ and $M =_\beta N$, then $\Gamma \vdash_{\text{CDV}} N{:}\sigma$.
- $\Gamma \vdash_{\text{CDV}} M{:}\sigma$ and σ is tail-proper, if and only if M has a head-normal form.
- $\Gamma \vdash_{\text{CDV}} M{:}\sigma$ and ω does not occur in Γ and σ, if and only if M has a normal form.

Normalised types play an important role in these characterisation properties.

The second type assignment system presented in [22] is a restricted version of the first, by restricting to normalised types and limiting the possible contexts that can be used in a derivation; therefore, it is not a proper extension of Curry's system: if $\Gamma \vdash M{:}\phi$ and x does not occur in Γ, then for $\lambda x.M$ only the type $\omega\to\phi$ can be derived. We present it here as a relevant system, i.e. a system that has only those statements in the context that are relevant to reach the conclusion; to have a consistent, inductive definition, we combine contexts in the rules, using intersection.

Definition 3.5 If $\Gamma_1,\ldots,\Gamma_n$ are contexts, then $\bigcap\{\Gamma_1,\ldots,\Gamma_n\}$ is the context defined as follows: $x{:}\sigma_1\cap\cdots\cap\sigma_m \in \bigcap\{\Gamma_1,\ldots,\Gamma_n\}$ if and only if $\{x{:}\phi_1,\ldots,x{:}\phi_m\}$ is the set of all statements about x that occur in the set $\Gamma_1 \cup \ldots \cup \Gamma_n$, and x occurs in this set.

We write $\bigcap_n \Gamma_i$ for $\bigcap\{\Gamma_1,\ldots,\Gamma_n\}$.

Definition 3.6 (c.f. [22]) *Restricted type assignment* is defined by:

$$(Ax): \dfrac{}{x{:}\phi \vdash x:\phi} \qquad (\omega): \dfrac{}{\vdash M{:}\omega} \qquad (\to I): \dfrac{\Gamma, x{:}\sigma \vdash M{:}\psi}{\Gamma \vdash \lambda x.M{:}\sigma\to\psi} \quad \dfrac{\Gamma \vdash M{:}\psi}{\Gamma \vdash \lambda x.M{:}\omega\to\psi}\ (a)$$

$$(\to E): \frac{\Gamma_1 \vdash M:\sigma\to\phi \quad \Gamma_2 \vdash N:\sigma}{\cap\{\Gamma_1,\Gamma_2\} \vdash MN:\phi} \qquad (\cap I): \frac{\Gamma_1 \vdash M:\phi_1 \quad \cdots \quad \Gamma_n \vdash M:\phi_n}{\cap_n\Gamma_i \vdash M:\phi_1\cap\cdots\cap\phi_n}$$

We write $\Gamma \vdash_{\text{R}} M:\sigma$ for statements derivable in this system.

(a): If x does not occur in Γ.

Again, the original natural deduction formulation of rule $(\to I)$ had the side-condition:

" If $\sigma = \cap_n\sigma_i$, and $x{:}\sigma_1,\ldots,x{:}\sigma_n$ are all and nothing but the statements about x on which $M:\psi$ depends; if $n = 0$, so in the derivation for $M:\psi$ there is no premise whose subject is x, then $\cap_n\sigma_i = \omega$ "

Since the context is relevant, this is implied in the above definition.

Notice that, in rule $(\to I)$, only those types actually used in the derivation can be abstracted. This implies that, for example, for the λ-term $\lambda ab.a$ the type $\phi\to\psi\to\phi$ cannot be derived, only types like $\phi\to\omega\to\phi$ can.

Properties of '$\vdash_{\text{R}}$' proven in [22] are:

- If $\Gamma \vdash_{\text{R}} M:\sigma$, then σ is a normalised type.
- If $\Gamma \vdash_{\text{R}} M:\sigma$ and $M \to_\eta N$, then $\Gamma \vdash_{\text{R}} N:\sigma$.

It is obvious that $\Gamma \vdash_{\text{R}} M:\phi$ implies $\Gamma \vdash_{\text{CDV}} M:\phi$, and that the converse does not hold, since $\vdash_{\text{CDV}} \lambda ab.a:\phi\to\psi\to\phi$. Moreover, type assignment in the unrestricted system is not invariant under η-reduction. For example, in '$\vdash_{\text{CDV}}$' we can derive

$$\cfrac{\cfrac{\cfrac{\cfrac{}{x:\sigma\to\phi,y:\sigma\cap\rho \vdash x:\sigma\to\phi} \qquad \cfrac{}{x:\sigma\to\phi,y:\sigma\cap\rho \vdash y:\sigma}\,(\cap E)}{x:\sigma\to\phi,y:\sigma\cap\rho \vdash xy:\phi}\,(\to E)}{x:\sigma\to\phi \vdash \lambda y.xy:\sigma\cap\rho\to\phi}\,(\to I)}{\emptyset \vdash \lambda xy.xy:(\sigma\to\phi)\to\sigma\cap\rho\to\phi}\,(\to I)$$

but we cannot derive $\emptyset \vdash_{\text{R}} \lambda x.x:(\sigma\to\phi)\to\sigma\cap\rho\to\phi$; this is due to the fact that, in '$\vdash_{\text{R}}$', there is no way to obtain $x{:}\sigma\cap\rho\to\phi$ from $x{:}\sigma\to\phi$. In '$\vdash_{\text{R}}$', in the $(\to I)$-rule, only types actually used for a term-variable can be used; the above use of rule $(\cap E)$ is there not allowed, forcing the above derivation to become:

$$\cfrac{\cfrac{\cfrac{\cfrac{}{x:\sigma\to\phi \vdash x:\sigma\to\phi} \qquad \cfrac{}{y:\sigma \vdash y:\sigma}}{x:\sigma\to\phi,y:\sigma \vdash xy:\phi}\,(\to E)}{x:\sigma\to\phi \vdash \lambda y.xy:\sigma\to\phi}\,(\to I)}{\emptyset \vdash \lambda xy.xy:(\sigma\to\phi)\to\sigma\to\phi}\,(\to I)$$

The closure under η-reduction, therefore, holds for '$\vdash_{\text{R}}$'.

3.4 A system with principal pairs

[21] also presents a system for which the principal pair property

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

is shown. The technique used for the proof of this property is very different for the one used for Curry's system, where unification [53] is the main tool to type an application [36], and type-substitution is the only operation used for the generation of pairs.

The principal type scheme for a term is in [21] (as well as in [55], and [6, 7], see Sections 3.6, 5.1, and 10) studied through the approximants of a term: terms with a finite number of approximants have a single principal pair, while terms with a infinite number of approximants have infinitely many 'principal pairs'. This can be understood from the observation that, using intersection types, also terms that do not have a normal form, but do have a head-normal form, or, in other words, terms that have an 'infinite' normal form, have types; in fact, they have infinitely many, inherently different types.

Example 3.7 Take, for example, the term $Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$. The set of approximants of the term is infinite (as is its Böhm tree):

$$\mathcal{A}(Y) \;=\; \{\bot\} \cup \{\lambda f.f^n\bot \mid n \geq 1\}$$

(where $f^0\bot = \bot$, and $f^n\bot = f(f^{n-1}\bot)$). It is easy to check that the elements of this set can be typed as follows:

$$\begin{array}{l}
\vdash \bot : \omega \\
\vdash \lambda f.f^1\bot : (\omega\to\phi)\to\phi \\
\vdash \lambda f.f^2\bot : (\omega\to\phi_2)\cap(\phi_2\to\phi_1)\to\phi_1 \\
\quad\vdots \\
\vdash \lambda f.f^n\bot : (\omega\to\phi_n)\cap(\phi_n\to\phi_{n-1})\cap\cdots\cap(\phi_2\to\phi_1)\to\phi_1 \\
\quad\vdots
\end{array}$$

For example (where $\Gamma = f{:}(\omega\to\phi_2)\cap(\phi_2\to\phi_1)$):

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash f:\phi_2\to\phi_1} \qquad \dfrac{\dfrac{}{\Gamma\vdash f:\omega\to\phi_2} \qquad \dfrac{}{\Gamma\vdash\bot:\omega}\,(\omega)}{\Gamma\vdash f\bot:\phi_2}\,(\to E)}{\Gamma\vdash f(f\bot):\phi_1}\,(\to E)}{\vdash \lambda f.f(f\bot):(\omega\to\phi_2)\cap(\phi_2\to\phi_1)\to\phi_1}\,(\to I)$$

Notice that, for each $n \leq m$, we can generate $(\omega\to\phi_n)\cap(\phi_n\to\phi_{n-1})\cap\cdots\cap(\phi_2\to\phi_1)\to\phi_1$ from $(\omega\to\phi_m)\cap(\phi_m\to\phi_{m-1})\cap\cdots\cap(\phi_2\to\phi_1)\to\phi_1$ using substitution, but cannot do so for the reverse using the operations provided below. Therefore, a functional characterisation of these terms, through a principal pair, cannot be represented in a finite way.

The type assignment system studied in [21] is the restricted one from [22], but with the set of types of the unrestricted system. The reason to not use the normalised types as well is the fact that ω is treated as a type constant; since ω can be substituted for φ in $\sigma\to\varphi$, also $\sigma\to\omega$ is considered a type.

That paper's main contribution is the proof of the principal pair property; the generation of 'instances' of the principal pair is done via the operation of substitution, which is standard and replaces type variables by types, and the newly defined operation of expansion (cf. Section 10.4).

The definition of expansion is rather complicated and deals with the replacement of a (sub)type 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, and the types in the conclusion and in the context of the subderivation will be instantiated into a number of copies.

Remark 3.8 For an illustration, suppose the following is a correct derivation:

$$\dfrac{\dfrac{}{\Gamma,x{:}\sigma{\to}\phi \vdash x:\sigma{\to}\phi} \qquad \begin{array}{c}\boxed{\qquad}\\ \Gamma \vdash N:\sigma\end{array}}{\Gamma,x{:}\sigma{\to}\phi \vdash xN:\phi}\,(\to E)$$

then, in general, the expansion that replaces σ by $\cap_n\sigma_i$ creates the following derivation:

$$\dfrac{\dfrac{}{\Gamma',x{:}\cap_n\sigma_i{\to}\phi \vdash x:\cap_n\sigma_i{\to}\phi} \qquad \dfrac{\begin{array}{c}\boxed{\qquad}\\ \Gamma' \vdash N:\sigma_1\end{array} \quad \cdots \quad \begin{array}{c}\boxed{\qquad}\\ \Gamma' \vdash N:\sigma_n\end{array}}{\Gamma' \vdash N:\cap_n\sigma_i}\,(\cap I)}{\Gamma',x{:}\cap_n\sigma_i{\to}\phi \vdash xN:\phi}\,(\to E)$$

An expansion indicates not only the type to be expanded, but also the number of copies that has to be generated, and possibly affects more types than just the one to be expanded. To clarify this, consider the following: suppose that μ is a subtype of σ that is to be expanded into n copies $\mu_1, \ldots, \mu_n$. If $\tau{\to}\mu$ is also a subtype of σ, then there are several ways to create the expansion of $\tau{\to}\mu$: just replacing μ by an intersection of copies of μ would generate $\tau{\to}\mu_1{\cap}\cdots{\cap}\mu_n$, which is not a correct type in '$\vdash_{\text{CDV}}$'. It could be replaced by $\cap_n(\tau{\to}\mu_i)$, but such an operation of expansion would not be *complete*, a property that is needed in the proof that the operations are sufficient (see Section 10.7). The subtype $\tau{\to}\mu$ will therefore be expanded into $\cap_n(\tau_i{\to}\mu_i)$, where the τ_i are copies of τ. Then τ is *affected* by the expansion of μ; all other occurrences elsewhere of τ should be expanded into $\cap_n\tau_i$, 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 subtypes 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 subtypes found.

The construction of the principal pair property follows the outline of Section 10; the technique used to show this property is the same as later used in [55] and [6]: it defines principal pairs of context 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 generalising this result to arbitrary λ-terms.

The set of ground pairs for a term $A \in \mathcal{A}$, as defined in [21], is proven to be complete for A, in the sense that all other pairs for A (i.e. pairs $\langle\Gamma,\sigma\rangle$ such that $\Gamma \vdash A:\sigma$) 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 $\Gamma \vdash A:\sigma$ with $A \in \mathcal{A}$, then there is a substitution S and a ground pair $\langle\Gamma',\sigma'\rangle$ for A such that $S(\langle\Gamma',\sigma'\rangle) = \langle\Gamma,\sigma\rangle$.
- If $\langle\Gamma,\sigma\rangle$ is a ground pair for $A \in \mathcal{A}$ and $\langle\Gamma',\sigma'\rangle$ can be obtained from $\langle\Gamma,\sigma\rangle$ by an expansion, then $\langle\Gamma',\sigma'\rangle$ is a ground pair for A.
- For all $A \in \mathcal{A}$, every ground pair for A is complete for A.

In the construction of principal pairs for λ-terms, first, for every $A \in \mathcal{A}$, a particular pair $pp(A)$ is chosen of context Π and type π (exactly the same as in Definition 10.6), called the *principal context scheme* and *principal type scheme* of A, respectively. This pair is called the *principal pair* of A.

The proof is completed by proving:

- $pp(A)$ is a ground pair for A.

- The *Approximation Theorem*: $\Gamma \vdash_{\text{CDV}} M{:}\sigma$ if and only if there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\text{CDV}} A{:}\sigma$ (cf. Theorem 6.9).
- $\{pp\,(A) \mid A \in \mathcal{A}(M)\}$ is complete for M.

Example 3.9 Notice that, by the Approximation Theorem, we can derive the types given in Example 3.7 for the approximants of Y for Y as well; for example:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{f{:}\omega\to\phi,x{:}\omega \vdash f:\omega\to\phi} \qquad \dfrac{}{f{:}\omega\to\phi,x{:}\omega\vdash xx:\omega}\,(\omega)}{f{:}\omega\to\phi,x{:}\omega\vdash f(xx):\phi}\,(\to E)}{f{:}\omega\to\phi\vdash \lambda x.f(xx):\omega\to\phi}\,(\to I) \qquad \dfrac{}{f{:}\omega\to\phi\vdash\lambda x.f(xx):\omega}\,(\omega)}{f{:}\omega\to\phi\vdash(\lambda x.f(xx))(\lambda x.f(xx)):\phi}\,(\to E)}{\vdash \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)):(\omega\to\phi)\to\phi}\,(\to I)$$

Take $\Gamma = f{:}(\omega\to\phi_2)\cap(\phi_2\to\phi_1)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma,x{:}\omega\to\phi_2\vdash f:\phi_2\to\phi_1} \qquad \begin{array}{c}\dfrac{\dfrac{}{\Gamma,x{:}\omega\to\phi_2\vdash x:\omega\to\phi_2} \qquad \dfrac{}{\Gamma,x{:}\omega\to\phi_2\vdash x:\omega}\,(\omega)}{\Gamma,x{:}\omega\to\phi_2\vdash xx:\phi_2}\,(\to E)\\ \vdots\end{array}}{\Gamma,x{:}\omega\to\phi_2\vdash f(xx):\phi_1}\,(\to E)}{\Gamma\vdash\lambda x.f(xx):(\omega\to\phi_2)\to\phi_1}\,(\to I) \qquad \dfrac{\dfrac{\dfrac{}{\Gamma,x{:}\omega\vdash f:\omega\to\phi_2} \qquad \dfrac{}{\Gamma,x{:}\omega\vdash xx:\omega}\,(\omega)}{\Gamma,x{:}\omega\vdash f(xx):\phi_2}\,(\to E)}{\Gamma\vdash\lambda x.f(xx):\omega\to\phi_2}\,(\to I)}{\Gamma\vdash(\lambda x.f(xx))(\lambda x.f(xx)):\phi_1}\,(\to E)}{\vdash\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)):(\omega\to\phi_2)\cap(\phi_2\to\phi_1)\to\phi_1}\,(\to I)$$

etc.

3.5 The Barendregt-Coppo-Dezani system

The type assignment system presented by Barendregt, Coppo and Dezani-Ciancaglini in [15] (the BCD-system) is based on '$\vdash_{\text{CDV}}$'. It strengthened that system by treating '$\cap$' as a normal type constructors like '$\to$', allowing intersections to occur at the right of arrow types. It also introduces a partial order relation '$\leq$' on types, adds the type assignment rule $(\leq)$ to close type assignment for this relation, and introduces a more general form of the rules concerning intersection. The set of types derivable for a λ-term in the extended system is a filter, i.e. a set closed under intersection and right-closed for '$\leq$'. The interpretation of a λ-term M by the set of types derivable for it – $[\![M]\!]_\xi$ – gives a filter λ-model $\mathcal{F}_{\text{BCD}}$[8].

The main contribution of [15] was to show completeness of type assignment, for which the type inclusion relation '$\leq$' and the type assignment rule $(\leq)$ were added. This is achieved by showing the statement: $\Gamma \vdash_{\text{BCD}} M{:}\sigma \iff [\![M]\!]_{\xi_\Gamma} \in v(\sigma)$ (the interpretation of the term M is an element of the interpretation of the type σ, if and only if M is typeable with σ), where $v : \mathcal{T}_{\text{BCD}} \to \mathcal{F}_{\text{BCD}}$ is a simple type interpretation as defined in [38]. In order to prove the '$\Leftarrow$'-part of this statement (completeness), the relation '$\leq$' is needed.

Definition 3.10 *i*) $\mathcal{T}_{\text{BCD}}$, the set of types in [15] is defined by:

$$\sigma,\tau \;::=\; \varphi \mid \omega \mid \sigma\to\tau \mid \sigma\cap\tau$$

ii) On $\mathcal{T}_{\text{BCD}}$, the type inclusion relation '$\leq$' is defined as the smallest pre-order such that:

[8] Called $\mathcal{F}$ in [15], but subscripted here to be able to distinguish it from other filter models.

$$\begin{array}{rclcrcl} \sigma & \leq & \omega & \quad & (\sigma\to\tau)\cap(\sigma\to\rho) & \leq & \sigma\to\tau\cap\rho \\ \sigma\cap\tau & \leq & \sigma,\tau & & \sigma\leq\tau \ \&\ \sigma\leq\rho & \Rightarrow & \sigma\leq\tau\cap\rho \\ \omega & \leq & \omega\to\omega & & \rho\leq\sigma \ \&\ \tau\leq\phi & \Rightarrow & \sigma\to\tau\leq\rho\to\phi \end{array}$$

iii) '$\sim$' is the equivalence relation induced by '$\leq$': $\sigma\sim\tau \iff \sigma\leq\tau\leq\sigma$.

iv) Contexts are defined as before, and $\Gamma\leq\Gamma'$ if and only if for every $x{:}\sigma' \in \Gamma'$ there is an $x{:}\sigma \in \Gamma$ such that $\sigma\leq\sigma'$, and $\Gamma\sim\Gamma' \iff \Gamma\leq\Gamma'\leq\Gamma$.

$\mathcal{T}_{\text{BCD}}$ may be considered modulo '$\sim$'. Then '$\leq$' becomes a partial order; notice that, if $\Gamma\subseteq\Gamma'$, then $\Gamma'\leq\Gamma$.

Definition 3.11 (c.f. [15]) *Type assignment* is defined by the following sequent-style natural deduction system.

$$\begin{array}{llll}
(\to I): \dfrac{\Gamma,x{:}\sigma \vdash M{:}\tau}{\Gamma \vdash \lambda x.M{:}\sigma\to\tau} & & (\to E): \dfrac{\Gamma \vdash M{:}\sigma\to\tau \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash MN{:}\tau} \\[3ex]
(\cap I): \dfrac{\Gamma \vdash M{:}\sigma \quad \Gamma \vdash M{:}\tau}{\Gamma \vdash M{:}\sigma\cap\tau} & & (\cap E): \dfrac{\Gamma \vdash M{:}\sigma\cap\tau}{\Gamma \vdash M{:}\sigma} \quad \dfrac{\Gamma \vdash M{:}\sigma\cap\tau}{\Gamma \vdash M{:}\tau} \\[3ex]
(\leq): \dfrac{\Gamma \vdash M{:}\sigma}{\Gamma \vdash M{:}\tau}\,(\sigma\leq\tau) & & (\omega): \dfrac{}{\Gamma \vdash M{:}\omega}
\end{array}$$

We will write $\Gamma \vdash_{\text{BCD}} M{:}\sigma$ for statements that are derived using these rules.

In '$\vdash_{\text{BCD}}$', there are several ways to deduce a desired result, due to the presence of the derivation rules $(\cap I)$, $(\cap E)$ and $(\leq)$, which allow superfluous steps in derivations; notice that $(\cap E)$ is included in $(\leq)$. In the CDV-systems, as in '$\vdash_{\text{S}}$' (Section 5) and '$\vdash_{\text{E}}$' (Section 4), these rules are not present and there is a one-to-one relationship between terms and skeletons of derivations. In other words: those systems are syntax directed.

An advantage of the presentation in [15] 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.

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

Definition 3.12 *i*) A *BCD-filter* is a subset $d \subseteq \mathcal{T}_{\text{BCD}}$ such that:

a) $\omega \in d$.

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

c) $\sigma\leq\tau \ \&\ \sigma \in d \Rightarrow \tau \in d$.

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

iii) For $d_1,d_2 \in \mathcal{F}_{\text{BCD}}$, we define application on BCD-filters by: $d_1 \cdot d_2 = \{\tau \in \mathcal{T}_{\text{BCD}} \mid \exists \sigma \in d_2\ [\sigma\to\tau \in d_1]\}$.

In constructing a complete system, the semantics of types plays a crucial role: the goal is to show that $\Gamma \vdash_{\text{E}} M{:}\sigma \iff \Gamma \models_{\text{s}} M{:}\sigma$; the latter represents a relation between the semantic interpretation of Γ, M and σ, so requires a type interpretation. As in [31, 49], and essentially following [37], 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 different notions of semantic satisfiability.

Definition 3.13 (Type interpretation) *i*) Let $\langle \mathcal{D}, \cdot, \varepsilon \rangle$ be a continuous λ-model. A mapping $v : \mathcal{T} \rightarrow \wp(\mathcal{D}) = \{X \,|\, X \subseteq \mathcal{D}\}$ is an *inference type interpretation* if and only if:

a) $\{\varepsilon \cdot d \,|\, \forall e \in v(\sigma)\ [d \cdot e \in v(\tau)]\} \subseteq v(\sigma \rightarrow \tau)$.

b) $v(\sigma \rightarrow \tau) \subseteq \{d \,|\, \forall e \in v(\sigma)\ [d \cdot e \in v(\tau)]\}$.

c) $v(\sigma \cap \tau) = v(\sigma) \cap v(\tau)$.

ii) Following [38], a type interpretation is *simple* if also:

$$v(\sigma \rightarrow \tau) = \{d \,|\, \forall e \in v(\sigma)\ [d \cdot e \in v(\tau)]\}.$$

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

$$v(\sigma \rightarrow \tau) = \{\varepsilon \cdot d \,|\, \forall e \in v(\sigma)\ [d \cdot e \in v(\tau)]\}.$$

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 3.14 (Satisfiability) Let $\mathcal{M} = \langle \mathcal{D}, \cdot, [\![\cdot]\!]^{\mathcal{M}}_{\cdot} \rangle$ be a λ-model and ξ a valuation of term-variables in $\mathcal{D}$, and v a type interpretation. We define $\vDash$ by;

i) $\mathcal{M}, \xi, v \vDash M : \sigma \iff [\![M]\!]^{\mathcal{M}}_{\xi} \in v(\sigma)$.

ii) $\mathcal{M}, \xi, v \vDash \Gamma \iff \mathcal{M}, \xi, v \vDash x{:}\sigma$ for every $x{:}\sigma \in \Gamma$.

iii) *a*) $\Gamma \vDash M : \sigma \iff \forall \mathcal{M}, \xi, v\ [\mathcal{M}, \xi, v \vDash \Gamma \Rightarrow \mathcal{M}, \xi, v \vDash M : \sigma]$.

b) $\Gamma \vDash_{\mathrm{s}} M : \sigma \iff \forall \mathcal{M}, \xi,$ *simple type interpretations* $v\ [\mathcal{M}, \xi, v \vDash \Gamma \Rightarrow \mathcal{M}, \xi, v \vDash M : \sigma]$.

c) $\Gamma \vDash_{\mathrm{F}} M : \sigma \iff \forall \mathcal{M}, \xi,$ *F type interpretations* $v\ [\mathcal{M}, \xi, v \vDash \Gamma \Rightarrow \mathcal{M}, \xi, v \vDash M : \sigma]$.

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

The following properties are proven in [15]:

- For all $M \in \Lambda$, $\{\sigma \,|\, \exists \Gamma\ [\Gamma \vdash_{\mathrm{BCD}} M{:}\sigma]\} \in \mathcal{F}_{\mathrm{BCD}}$.
- Let ξ be a valuation of term-variables in $\mathcal{F}_{\mathrm{BCD}}$ and $\Gamma_{\xi} = \{x{:}\sigma \,|\, \sigma \in \xi(x)\}$. For $M \in \Lambda$, define the interpretation of M in $\mathcal{F}_{\mathrm{BCD}}$ via $[\![M]\!]_{\xi} = \{\sigma \,|\, \Gamma_{\xi} \vdash_{\mathrm{BCD}} M{:}\sigma\}$. Using the method of [40] it is shown that $\langle \mathcal{F}_{\mathrm{BCD}}, \cdot, [\![\cdot]\!] \rangle$ is a λ-model.

The main result of [15] is obtained by proving:

Property 3.15 i) Soundness: $\Gamma \vdash_{\mathrm{BCD}} M{:}\sigma \Rightarrow \Gamma \vDash_{\mathrm{s}} M{:}\sigma$.

ii) Completeness: $\Gamma \vDash_{\mathrm{s}} M{:}\sigma \Rightarrow \Gamma \vdash_{\mathrm{BCD}} M{:}\sigma$.

The proof of completeness is obtained in a way very similar to that of Theorem 7.13. The results of [15] in fact show that type assignment in '$\vdash_{\mathrm{BCD}}$' is complete with respect to *simple* type semantics; this in contrast to '$\vdash_{\mathrm{s}}$' (presented in [3], see also Section 5), that is complete with respect to the inference semantics.

As is shown by the results achieved for '$\vdash_{\mathrm{E}}$' in Section 7, allowing intersections on the right of arrow types is, in fact, not needed to solve the problem of completeness of type assignment (see Property 3.15): the mere introduction of a relation on normalised types with contra-variance in the arrow would have done the trick.

A characterisation of strong normalisation is shown in [3], by proving that the set of all terms typeable by means of the derivation rules $(\cap I), (\cap E), (\rightarrow I)$ and $(\rightarrow E)$ of '$\vdash_{\mathrm{BCD}}$' is exactly the set of strongly normalisable terms. The proof for this property in that paper needs the rule $(\leq)$ for the contra-variant '$\leq$'-relation. In [10], an alternative proof of strong normalisation was given (much like that presented in Section 9 that appeared in [11]), but for the strict type assignment system of [3] (see Section 5); this proof does not need a contra-variant '$\leq$'-relation.

3.6 Principal pairs for '$\vdash_{\text{BCD}}$'

For '$\vdash_{\text{BCD}}$', principal type schemes are defined by [55]. 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 that paper, all constructions and definitions are made modulo the equivalence relation $\sim$. In fact, the complexity inserted in BCD-types, by allowing for intersection types on the right of the arrow type constructor, disturbs greatly the complexity of the definition of expansion, and, through that, the accessibility of that paper.

The first operation defined is substitution, that is defined without restriction: the type that is to be substituted can be every element of $\mathcal{T}_{\text{BCD}}$. Next, the operation of expansion is defined, which is a generalisation of the notion of expansion defined in [21] in Section 3.4; since intersections now also can appear on the right of an arrow, the construction of the set of types affected by an expansion is more involved. Both substitution and expansions are in the natural way extended to operations on contexts 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 for approximants in the following sense:

- (Soundness) Assume $\Gamma \vdash_{\text{BCD}} A{:}\sigma$, and let O be an operation of substitution, expansion or rise, and $O(\langle \Gamma,\sigma \rangle) = \langle \Gamma',\sigma' \rangle$, then $\Gamma' \vdash_{\text{BCD}} A{:}\sigma'$.

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. Principal pairs are defined for terms in $\lambda\beta$-normal form, in exactly the same way as in Definition 10.6. To come to the proof of completeness of these operations, [55] first proves the approximation theorem for '$\vdash_{\text{BCD}}$':

Property 3.16 *$\Gamma \vdash_{\text{BCD}} M{:}\sigma$ iff there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\text{BCD}} A{:}\sigma$.*

which is used in the proof of the principal type property, as in Theorem 10.30. The proof is completed by proving first that, when $\mathcal{A}(M)$ is finite, then $\{pp(A) \mid A \in \mathcal{A}(M)\}$ has a maximal element (see Theorem 10.28 and Definition 10.29). Then the following is proven:

Property 3.17 *i) Let $A \in \mathcal{A}$, then for any pair $\langle \Gamma,\sigma \rangle$ such that $\Gamma \vdash_{\text{BCD}} A{:}\sigma$ there exists a linear chain Ch such that $Ch(pp(A)) = \langle \Gamma,\sigma \rangle$.*

ii) Let $\Gamma \vdash_{\text{BCD}} M{:}\sigma$.

a) $\mathcal{A}(M)$ is finite. Then $\{pp(A) \mid A \in \mathcal{A}(M)\}$ has a maximal element, say $\langle \Pi,\pi \rangle$. Then there exists a linear chain Ch such that $Ch(\langle \Pi,\pi \rangle) = \langle \Gamma,\sigma \rangle$.

b) $\mathcal{A}(M)$ is infinite. Then there exist a pair $\langle \Pi,\pi \rangle \in \{pp(A) \mid A \in \mathcal{A}(M)\}$ and a linear chain Ch, such that $Ch(\langle \Pi,\pi \rangle) = \langle \Gamma,\sigma \rangle$.

4 Essential Intersection type assignment

In this section, we will present the *essential intersection system*, a notion of type assignment system that is a restricted version of '$\vdash_{\text{BCD}}$' as first presented in [7], together with some of its properties. The major feature of this restricted system is, compared to '$\vdash_{\text{BCD}}$', 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 '$\vdash_{\mathrm{S}}$' that was presented in [3] (see Section 5); the main difference is that '$\vdash_{\mathrm{S}}$' is not closed for η-reduction, whereas '$\vdash_{\mathrm{E}}$' is.

Strict types are the types that are strictly needed to assign a type to a term that is non-trivially typeable in '$\vdash_{\mathrm{BCD}}$'. In the set of strict types, intersection type schemes and the type constant ω play a limited role. In particular, intersection type schemes (so also ω) occur in strict types only as subtype at the left-hand side of an arrow type scheme, as in the types of [20, 21, 22]. Moreover, ω is taken to be the empty intersection: if $n=0$, then $\phi_1\cap\cdots\cap\phi_n \equiv \omega$, so ω does not occur in an intersection subtype or on the right of an arrow[9].

Definition 4.1 (Strict types) *i*) The set $\mathcal{T}$ of *(essential) intersection types*, that is ranged over by $\sigma,\tau,\ldots$, and its subset $\mathcal{T}_{\mathrm{s}}$ of *strict types*, ranged over by $\phi,\psi,\ldots$, are defined through the grammar:

$$\begin{array}{rcl} \phi,\psi & ::= & \varphi \mid \sigma{\rightarrow}\psi \\ \sigma,\tau & ::= & \phi_1\cap\cdots\cap\phi_n \quad (n\geq 0) \end{array}$$

with φ ranging over $\mathcal{V}$, the set of type variables.

ii) We define ω as the empty intersection, so if $\sigma=\phi_1\cap\cdots\cap\phi_n$ with $n\geq 0$, then $\phi_i \neq \omega$ for all $i\in \underline{n}$.

iii) The relation '$\leq$' is defined as the least pre-order on $\mathcal{T}$ such that:

$$\begin{array}{rcl} \phi_1\cap\cdots\cap\phi_n & \leq_{\mathrm{E}} & \phi_i, \text{ for all } i\in\underline{n},\ n\geq 1 \\ \tau\leq_{\mathrm{E}}\phi_i, \text{ for all } i\in\underline{n} & \Rightarrow & \tau\leq_{\mathrm{E}}\phi_1\cap\cdots\cap\phi_n,\ n\geq 0 \\ \rho\leq_{\mathrm{E}}\sigma \ \&\ \phi\leq_{\mathrm{E}}\psi & \Rightarrow & \sigma{\rightarrow}\phi\leq_{\mathrm{E}}\rho{\rightarrow}\psi \end{array}$$

iv) On $\mathcal{T}$, the relation '$\sim_{\mathrm{E}}$' is defined by: $\sigma\sim_{\mathrm{E}}\tau \iff \sigma\leq_{\mathrm{E}}\tau\leq_{\mathrm{E}}\sigma$.

v) The relations '$\leq_{\mathrm{E}}$' and '$\sim_{\mathrm{E}}$', are, as in Definition 3.10, extended to contexts.

Notice that, by the second clause of part (*iii*), $\sigma\leq_{\mathrm{E}}\omega$, for all σ.

Compared to types defined previously, the set of types as considered in [20] are ω-free strict types (see Definition 8.1); that in [22] treats ω as a type constant rather than the empty intersection (Definition 3.3), so strict types are a restriction of that set. Compared to Definition 3.4, normalising types is just the same as treating ω as an empty intersection, as well as not allowing ω to appear on the right-hand side of an arrow. In fact, the set of normalised types coincides with the set of strict types and normalised intersection types correspond to strict intersection types.

Since it is easy to show that intersection is commutative ($\sigma\cap\tau\sim_{\mathrm{E}}\tau\cap\sigma$) and associative ($\sigma\cap(\tau\cap\rho)\sim_{\mathrm{E}}(\sigma\cap\tau)\cap\rho$), we can arbitrarily swap the order of subtypes in an intersection, and will write $\phi_1\cap\cdots\cap\phi_n$ for the type $\phi_1\cap\cdots\cap\phi_n$, and $\cap_1\phi_i=\phi_1$.

Lemma 4.2 For the relation '$\leq_{\mathrm{E}}$', the following properties hold:

$$\begin{array}{rcl} \varphi\leq_{\mathrm{E}}\phi & \iff & \phi\equiv\varphi. \\ \omega\leq_{\mathrm{E}}\sigma & \iff & \sigma\equiv\omega. \\ \sigma{\rightarrow}\phi\leq_{\mathrm{E}}\rho\in\mathcal{T}_{\mathrm{s}} & \iff & \rho\equiv\mu{\rightarrow}\psi \ \&\ \mu\leq_{\mathrm{E}}\sigma \ \&\ \phi\leq_{\mathrm{E}}\psi \\ \phi_1\cap\cdots\cap\phi_n\leq_{\mathrm{E}}\tau\in\mathcal{T}_{\mathrm{s}} & \Rightarrow & \exists\, i\in\underline{n}\ [\phi_i\leq_{\mathrm{E}}\tau] \\ \sigma\leq_{\mathrm{E}}\tau & \Rightarrow & \sigma=\phi_1\cap\cdots\cap\phi_n \ \&\ \tau=\tau_1\cap\cdots\cap\tau_m \ \&\ \forall\, j\in\underline{m}\ \exists\, i\in\underline{n}\ [\phi_i\leq_{\mathrm{E}}\psi_j] \end{array}$$

Proof: Easy. ■

[9] It is worthwhile to notice that this definition of types would not be adequate in the setting of *lazy evaluation*, where an abstraction should always have an arrow type, even if it does not return a result: this forces the admission of $\sigma{\rightarrow}\omega$ as a type.

Definition 4.3 (Type assignment) *i*) *Essential intersection type assignment* and *derivations* are defined by the following natural deduction system:

$$(Ax): \frac{}{\Gamma,x{:}\sigma \vdash x{:}\phi}\ (\sigma \leq_{\rm E} \phi) \qquad (\cap I): \frac{\Gamma \vdash M{:}\phi_1 \quad \cdots \quad \Gamma \vdash M{:}\phi_n}{\Gamma \vdash M{:}\phi_1\cap\cdots\cap\phi_n}\ (n \geq 0)$$

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

ii) We write $\Gamma \vdash_{\rm E} M{:}\sigma$ if this statement is derivable using an essential intersection derivation, and $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\sigma$ when this result was obtained through the derivation $\mathcal{D}$.

The choice to treat ω as the empty intersection is motivated by the following observation. If we define $[\![\sigma]\!]$ to be the set of terms that can be assigned the type σ, then, for all $\sigma_1,\ldots,\sigma_n$:

$$[\![\phi_1\cap\cdots\cap\phi_n]\!] \subseteq [\![\phi_1\cap\cdots\cap\phi_{n-1}]\!] \subseteq \ldots \subseteq [\![\phi_1\cap\phi_2]\!] \subseteq [\![\phi_1]\!].$$

It is natural to extend this sequence with $[\![\phi_1]\!] \subseteq [\![\ ]\!]$, and therefore to define that the semantics of the empty intersection is the whole set of λ-terms; this is justified, since, via rule $(\cap I)$, we have $\Gamma \vdash_{\rm E} M{:}\omega$, for all Γ, M, exactly as in '$\vdash_{\rm BCD}$'.

Lemma 4.4 For this notion of type assignment, the following properties hold:

$$\begin{array}{lcl}
\Gamma \vdash_{\rm E} x{:}\sigma \ \&\ \sigma \neq \omega & \iff & \exists \rho \in \mathcal{T}\ [x{:}\rho \in \Gamma \ \&\ \rho \leq_{\rm E} \sigma] \\
\Gamma \vdash_{\rm E} MN{:}\phi & \iff & \exists \tau \in \mathcal{T}\ [\Gamma \vdash_{\rm E} M{:}\tau\to\phi \ \&\ \Gamma \vdash_{\rm E} N{:}\tau] \\
\Gamma \vdash_{\rm E} \lambda x.M{:}\psi & \iff & \exists \rho \in \mathcal{T}, \phi \in \mathcal{T}_{\rm s}\ [\psi = \rho\to\phi \ \&\ \Gamma,x{:}\rho \vdash_{\rm E} M{:}\phi] \\
\Gamma \vdash_{\rm E} M{:}\sigma \ \&\ \sigma \neq \omega & \iff & \exists \phi_1,\ldots,\phi_n\ [\sigma = \phi_1\cap\cdots\cap\phi_n \ \&\ \forall i \in \underline{n}\ [\Gamma \vdash_{\rm E} M{:}\phi_i]] \\
\Gamma \vdash_{\rm E} M{:}\sigma & \Rightarrow & \{x{:}\rho \mid x{:}\rho \in \Gamma \ \&\ x \in fv(M)\} \vdash_{\rm E} M{:}\sigma
\end{array}$$

Proof: Easy. ■

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

4.1 Eta reduction

Although the rule (Ax) is defined only for term-variables, '$\vdash_{\rm E}$' is closed for '$\leq_{\rm E}$' and weakening.

Lemma 4.5 If $\Gamma \vdash_{\rm E} M{:}\sigma$ and $\Gamma' \leq_{\rm E} \Gamma, \sigma \leq_{\rm E} \tau$, then $\Gamma \vdash_{\rm E} M{:}\tau$, so the following is an admissible rule in '$\vdash_{\rm E}$':

$$(\leq_{\rm E}): \frac{\Gamma \vdash M{:}\sigma}{\Gamma' \vdash M{:}\tau}\ (\Gamma' \leq_{\rm E} \Gamma, \sigma \leq_{\rm E} \tau)$$

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

(Ax): Then $M \equiv x$, and there is $x{:}\rho \in \Gamma$ such that $\rho \leq_{\rm E} \sigma$. Since $\Gamma' \leq_{\rm E} \Gamma$, there is $x{:}\mu \in \Gamma'$ such that $\mu \leq_{\rm E} \rho$. Notice that $\mu \leq_{\rm E} \rho \leq_{\rm E} \sigma \leq_{\rm E} \tau$, so, by Lemma 4.4, $\Gamma' \vdash_{\rm E} x{:}\tau$.

$(\to I)$: Then $M \equiv \lambda x.M'$, and there are ρ,ϕ such that $\sigma = \rho\to\phi$ and $\Gamma,x{:}\rho \vdash_{\rm E} M'{:}\phi$. By Lemma 4.2 there are $\rho_i,\mu_1,\ldots,\mu_n$ such that $\tau = \cap_n(\rho_i\to\phi_i)$, and for $i \in \underline{n}$, $\rho_i \leq_{\rm E} \rho$ and $\phi \leq_{\rm E} \phi_i$. Since $\Gamma' \leq_{\rm E} \Gamma$ and $\rho_i \leq_{\rm E} \rho$, also $\Gamma',x{:}\rho_i \leq_{\rm E} \Gamma,x{:}\rho$, and $\Gamma',x{:}\rho_i \vdash_{\rm E} M'{:}\phi_i$ by induction. So, by $(\to I)$, for every $i \in \underline{n}$, $\Gamma' \vdash_{\rm E} \lambda x.M'{:}\rho_i\to\phi_i$, so, by $(\cap I)$, $\Gamma' \vdash_{\rm E} \lambda x.M'{:}\tau$.

$(\to E)$: Then $M \equiv M_1M_2$, $\sigma = \phi$, and $\Gamma \vdash_E M_1 : \mu\to\phi$ and $\Gamma \vdash_E M_2 : \mu$ for some μ. Let $\tau = \tau_1\cap\cdots\cap\tau_n$, then, for all $i \in \underline{n}$, $\phi \leq \psi_i$, so also $\mu\to\phi \leq_E \mu\to\psi_i$ and, by induction, $\Gamma' \vdash_E M_1 : \mu\to\psi_i$. Then, by $(\to E)$, $\Gamma' \vdash_E M_1M_2 : \psi_i$ for all $i \in \underline{n}$, so $\Gamma' \vdash_E M_1M_2 : \tau$ by $(\cap I)$.

$(\cap I)$: Then $\sigma = \phi_1\cap\cdots\cap\phi_n$, and, for every $i \in \underline{n}$, $\Gamma \vdash_E M : \phi_i$. Let $\tau = \tau_1\cap\cdots\cap\tau_m$ then, for every $j \in \underline{m}$, there is an $i \in \underline{n}$ such that $\phi_i \leq_E \psi_j$. By induction, for every $j \in \underline{m}$, $\Gamma' \vdash_E M : \psi_j$. But then, by $(\cap I)$, $\Gamma' \vdash_E M : \tau$. ■

Now it is easy to prove that essential type assignment in this system is closed under η-reduction. The proof for this result is split in two parts, Lemma 4.6 and Theorem 4.7. The lemma is also used in the proof of Lemma 6.2.

Lemma 4.6 *If $\Gamma, x{:}\sigma \vdash_E Mx : \phi$ and $x \notin fv(M)$ then $\Gamma \vdash_E M : \sigma\to\phi$.*

Proof:
$$\begin{array}{ll}
\Gamma, x{:}\sigma \vdash_E Mx : \phi \ \&\ x \notin fv(M) & \Rightarrow (\to E) \\
\exists \tau\, [\Gamma, x{:}\sigma \vdash_E M : \tau\to\phi \ \&\ \Gamma, x{:}\sigma \vdash_E x : \tau \ \&\ x \notin fv(M)] & \Rightarrow (4.4) \\
\exists \tau\, [\Gamma, x{:}\sigma \vdash_E M : \tau\to\phi \ \&\ \sigma \leq_E \tau \ \&\ x \notin fv(M)] & \Rightarrow (4.4) \\
\exists \tau\, [\Gamma \vdash_E M : \tau\to\phi \ \&\ \sigma \leq_E \tau] & \Rightarrow (4.1(\mathit{iii})) \\
\exists \tau\, [\Gamma \vdash_E M : \tau\to\phi \ \&\ \tau\to\phi \leq_E \sigma\to\phi] & \Rightarrow (4.5) \\
\Gamma \vdash_E M : \sigma\to\phi. \ ■ &
\end{array}$$

We can now show that '$\vdash_E$' is closed for η-reduction.

Theorem 4.7 ('$\vdash_E$' CLOSED FOR $\to_\eta$) *$\Gamma \vdash_E M : \sigma \ \&\ M \to_\eta N \Rightarrow \Gamma \vdash_E N : \sigma$.*

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

$(\sigma = \psi \in \mathcal{T}_s)$: Then:
$$\begin{array}{ll}
\Gamma \vdash_E \lambda x.Mx : \psi \ \&\ x \notin fv(M) & \Rightarrow (\to I) \\
\exists \rho,\phi\, [\psi = \rho\to\phi \ \&\ \Gamma, x{:}\rho \vdash_E Mx : \phi] & \Rightarrow (4.6) \\
\Gamma \vdash_E M : \psi. &
\end{array}$$

$(\sigma = \phi_1\cap\cdots\cap\phi_n)$: Then, by $(\cap I)$, $\Gamma \vdash_E \lambda x.Mx : \phi_i$ for all $i \in \underline{n}$, so, by the previous part, $\Gamma \vdash_E M : \phi_i$, so, by $(\cap I)$, $\Gamma \vdash_E M : \sigma$. ■

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

Types are not invariant under η-expansion; for example, we can derive $\vdash_E \lambda x.x : \phi\to\phi$, but not $\vdash_E \lambda xy.xy : \phi\to\phi$.

Example 4.8 Notice that $\lambda xy.xy \to_\eta \lambda x.x$; we can easily derive $\vdash_E \lambda xy.xy : (\sigma\to\phi)\to\sigma\cap\rho\to\phi$ and $\vdash_E \lambda x.x : (\sigma\to\phi)\to\sigma\cap\rho\to\phi$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}\sigma\to\phi, y{:}\sigma\cap\rho \vdash x : \sigma\to\phi} \qquad \dfrac{}{x{:}\sigma\to\phi, y{:}\sigma\cap\rho \vdash y : \sigma}\,(\sigma\cap\rho \leq_E \sigma)}{x{:}\sigma\to\phi, y{:}\sigma\cap\rho \vdash xy : \phi}\,(\to E)}{x{:}\sigma\to\phi \vdash \lambda y.xy : \sigma\cap\rho\to\phi}\,(\to I)}{\vdash \lambda xy.xy : (\sigma\to\phi)\to\sigma\cap\rho\to\phi}\,(\to I)$$

$$\dfrac{\dfrac{}{x{:}\sigma\to\phi \vdash x : \sigma\cap\rho\to\phi}\,(\sigma\to\phi \leq_E \sigma\cap\rho\to\phi)}{\vdash \lambda x.x : (\sigma\to\phi)\to\sigma\cap\rho\to\phi}\,(\to I)$$

4.2 Subject reduction and expansion

As in [21, 15], it is possible to prove that '$\vdash_{\mathrm{E}}$' system is closed under '$=_\beta$'. In the latter paper this result was obtained by building a filter λ-model; from the fact that every M is interpreted by the set of its assignable types, and that set is a filter, the result is then immediate (see also Corollary 7.12). We will here prove this result in the same way in Corollary 7.12, but first show it directly, without using a model; in this way the precise role of the type constructor '$\cap$' and the type constant ω can be made apparent.

Suppose first that $\Gamma \vdash_{\mathrm{E}} (\lambda x.M)N:\phi$. By $(\rightarrow E)$, there exists τ such that

$$\Gamma \vdash_{\mathrm{E}} \lambda x.M:\tau\rightarrow\phi \text{ and } \Gamma \vdash_{\mathrm{E}} N:\tau.$$

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

$$\Gamma, x{:}\tau \vdash_{\mathrm{E}} M:\phi \text{ and } \Gamma \vdash_{\mathrm{E}} N:\tau.$$

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

$$\frac{}{\Gamma, x{:}\tau \vdash x:\psi_j}\,(Ax)$$

and these are all the applications of rule (Ax) that deal with x. Thus, for all $j \in \underline{m}$, $\tau \leq_{\mathrm{E}} \psi_j$ and, by Lemma 4.5, $\Gamma \vdash_{\mathrm{E}} N:\psi_j$. Then a derivation for $\Gamma \vdash_{\mathrm{E}} M[N/x]:\phi$ can be obtained by replacing, for every $j \in \underline{m}$, in the derivation for $\Gamma, x{:}\tau \vdash_{\mathrm{E}} M:\phi$, the subderivation $\Gamma, x{:}\tau \vdash_{\mathrm{E}} x:\psi_j$ by the (new) derivation for $\Gamma \vdash_{\mathrm{E}} N:\psi_j$. (The operation described here is at the basis of derivation reduction as discussed in Section 9, and is formalised in Definition 9.4.)

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

$$\Gamma \vdash_{\mathrm{E}} M[N/x]:\psi \;\Rightarrow\; \Gamma \vdash_{\mathrm{E}} (\lambda x.M)N:\psi.$$

Assume that 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 $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\psi$, probably with several different types $\sigma_1,\ldots,\sigma_n$. A derivation for $\Gamma, x{:}\phi_1\cap\cdots\cap\phi_n \vdash_{\mathrm{E}} M:\psi$ can be obtained by replacing, in $\mathcal{D}$, all derivations for $\Gamma \vdash_{\mathrm{E}} N:\phi_i$ by the derivation for $x{:}\phi_1\cap\cdots\cap\phi_n \vdash_{\mathrm{E}} x:\phi_i$. Then, using $(\rightarrow I)$, $\Gamma \vdash_{\mathrm{E}} \lambda x.M:\phi_1\cap\cdots\cap\phi_n\rightarrow\psi$, and, using $(\cap I)$ on the collection of removed sub-derivations, $\Gamma \vdash_{\mathrm{E}} N:\phi_1\cap\cdots\cap\phi_n$. Then, using $(\rightarrow E)$, the redex can be typed.

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

To show this result formally, first a substitution lemma is proven. Notice that, unlike for many other notions of type assignment (Curry's system, [34]'s polymorphic type discipline, ML [27]), the implication holds in both directions.

Lemma 4.9 (Substitution lemma) $\exists \rho\, [\Gamma, x{:}\rho \vdash_{\mathrm{E}} M:\sigma \;\&\; \Gamma \vdash_{\mathrm{E}} N:\rho] \iff \Gamma \vdash_{\mathrm{E}} M[N/x]:\sigma$.

Proof: By induction on M. Only the case $\sigma = \phi \in \mathcal{T}_{\mathrm{S}}$ is considered.

$$\begin{array}{lll}
(M \equiv x):\ (\Rightarrow): & \exists \rho\, [\Gamma, x{:}\rho \vdash_{\mathrm{E}} x:\phi \;\&\; \Gamma \vdash_{\mathrm{E}} N:\rho] & \Rightarrow (4.4) \\
& \exists \rho\, [\rho \leq_{\mathrm{E}} \phi \;\&\; \Gamma \vdash_{\mathrm{E}} N:\rho] & \Rightarrow (4.5) \\
& \Gamma \vdash_{\mathrm{E}} N:\phi & \Rightarrow \\
& \Gamma \vdash_{\mathrm{E}} x[N/x]:\phi. &
\end{array}$$

$(\Leftarrow)$: $\Gamma \vdash_{\mathrm{E}} x[N/x]:\phi \Rightarrow \Gamma, x{:}\phi \vdash_{\mathrm{E}} x:\phi \;\&\; \Gamma \vdash_{\mathrm{E}} N:\phi$.

$(M \equiv y \neq x)$: $(\Rightarrow)$: By Lemma 4.4, since $y[N/x] \equiv y$, and $x \notin fv(y)$.

$(\Leftarrow)$: $\Gamma \vdash_{\mathrm{E}} y[N/x]:\phi \Rightarrow \Gamma \vdash_{\mathrm{E}} y:\phi$. Take $\rho = \omega$; by Lemma 4.5, $\Gamma, x{:}\omega \vdash_{\mathrm{E}} y:\phi$.

$$\begin{array}{ll}
(M \equiv \lambda y.M'):(\Leftrightarrow):\quad \exists \rho\,[\Gamma,x{:}\rho \vdash_{\rm E} \lambda y.M:\phi \;\&\; \Gamma \vdash_{\rm E} N:\rho] & \Leftrightarrow (\rightarrow I)\\
\quad \exists \rho,\mu,\psi\,[\Gamma,x{:}\rho,y{:}\mu \vdash_{\rm E} M:\psi \;\&\; \phi = \mu\rightarrow\psi \;\&\; \Gamma \vdash_{\rm E} N:\rho] & \Leftrightarrow (IH)\\
\quad \exists \mu,\psi\,[\Gamma,y{:}\mu \vdash_{\rm E} M[N/x]:\psi \;\&\; \phi = \mu\rightarrow\psi] & \Leftrightarrow (\rightarrow I)\\
\quad \Gamma \vdash_{\rm E} \lambda y.(M[N/x]):\phi & \Leftrightarrow\\
\quad \Gamma \vdash_{\rm E} (\lambda y.M)[N/x]:\phi. &
\end{array}$$

$$\begin{array}{ll}
(M \equiv M_1M_2):\ (\Rightarrow):\quad \exists \rho\,[\Gamma,x{:}\rho \vdash_{\rm E} M_1M_2:\phi \;\&\; \Gamma \vdash_{\rm E} N:\rho] & \Rightarrow (\rightarrow E)\\
\quad \exists \rho,\tau\,[\Gamma,x{:}\rho \vdash_{\rm E} M_1:\tau\rightarrow\phi \;\&\; \Gamma,x{:}\rho \vdash_{\rm E} M_2:\tau \;\&\; \Gamma \vdash_{\rm E} N:\rho] & \Rightarrow (IH)\\
\quad \exists \tau\,[\Gamma \vdash_{\rm E} M_1[N/x]:\tau\rightarrow\phi \;\&\; \Gamma \vdash_{\rm E} M_2[N/x]:\tau] & \Rightarrow (\rightarrow E)\\
\quad \Gamma \vdash_{\rm E} M_1[N/x]M_2[N/x]:\phi & \Rightarrow \Gamma \vdash_{\rm E} (M_1M_2)[N/x]:\phi
\end{array}$$

$$\begin{array}{ll}
(\Leftarrow):\ \Gamma \vdash_{\rm E} (M_1M_2)[N/x]:\phi & \Rightarrow\\
\quad \Gamma \vdash_{\rm E} M_1[N/x]M_2[N/x]:\phi & \Rightarrow (\rightarrow E)\\
\quad \exists \tau\,[\Gamma \vdash_{\rm E} M_1[N/x]:\tau\rightarrow\phi \;\&\; \Gamma \vdash_{\rm E} M_2[N/x]:\tau] & \Rightarrow (IH)\\
\quad \exists \rho_1,\rho_2,\tau\,[\Gamma,x{:}\rho_1 \vdash_{\rm E} M_1:\tau\rightarrow\phi \;\&\; \Gamma \vdash_{\rm E} N:\rho_1 \;\&\; \Gamma,x{:}\rho_2 \vdash_{\rm E} M_2:\tau \;\&\; \Gamma \vdash_{\rm E} N:\rho_2] & \\
 & \Rightarrow (\rho = \rho_1\cap\rho_2 \;\&\; (\cap I) \;\&\; 4.5)\\
\quad \exists \rho\,[\Gamma,x{:}\rho \vdash_{\rm E} M_1M_2:\phi \;\&\; \Gamma \vdash_{\rm E} N:\rho].\ \blacksquare &
\end{array}$$

Notice that '$\leq$' plays no role in this proof, other than its $(\cap E)$ behaviour, which means that this is also a proof for this property in the strict system of the next section.

This result leads immediately to the following:

Theorem 4.10 ('$\vdash_{\rm E}$' CLOSED FOR '$=_\beta$') *The following rules are admissible in '$\vdash_{\rm E}$':*

$$(cut):\ \frac{\Gamma,x{:}\rho \vdash M:\sigma \quad \Gamma \vdash N:\rho}{\Gamma \vdash M[N/x]:\sigma} \qquad (=_\beta):\ \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash N:\sigma}\ (M =_\beta N)$$

Proof: The first follows by the previous lemma; for the second, we reason by induction on the definition of '$=_\beta$'. We only show the part of a redex, in particular when typed with a strict type: $\Gamma \vdash_{\rm E} (\lambda x.M)N:\phi \Leftrightarrow \Gamma \vdash_{\rm E} M[N/x]:\phi$; all other cases follow by straightforward induction. To conclude, as argued above, if $\Gamma \vdash_{\rm E} (\lambda x.M)N:\phi$, then, by $(\rightarrow E)$ and $(\rightarrow I)$, there exists a ρ such that $\Gamma,x{:}\rho \vdash_{\rm E} M:\phi$ and $\Gamma \vdash_{\rm E} N:\rho$; the converse of this result holds, obviously, as well. The result then follows by applying Lemma 4.9. ■

5 The strict system

Another system (in fact, the first) that uses strict intersection types is the *strict* system of [3], in which the normalisation properties and completeness are shown, and for which strong normalisation of derivation reduction and characterisation results are proven with the same technique as used in Section 9 as was first shown in [10]. It is defined as follows:

Definition 5.1 The *strict type assignment* is defined by the following natural deduction system:

$$(\cap E):\ \frac{}{\Gamma,x{:}\phi_1\cap\cdots\cap\phi_n \vdash x:\phi_i}\ (n\geq 1, i\in \underline{n}) \qquad (\rightarrow I):\ \frac{\Gamma,x{:}\sigma \vdash M:\phi}{\Gamma \vdash \lambda x.M:\sigma\rightarrow\phi}$$

$$(\cap I):\ \frac{\Gamma \vdash M:\phi_1 \quad \cdots \quad \Gamma \vdash M:\phi_n}{\Gamma \vdash M:\phi_1\cap\cdots\cap\phi_n}\ (n\geq 0) \qquad (\rightarrow E):\ \frac{\Gamma \vdash M:\sigma\rightarrow\phi \quad \Gamma \vdash N:\sigma}{\Gamma \vdash MN:\phi}$$

We write $\Gamma \vdash_{\rm S} M:\sigma$ for statements derivable in this system.

We should emphasise the (only) difference between essential type assignment of Definition 4.3 and the strict one: instead of the rule $(\cap E)$ given above, it contains the rule (Ax), which uses a contra-variant type inclusion relation. Notice that rule $(\cap E)$ is a special case of rule (Ax) in that $\phi_1\cap\cdots\cap\phi_n \leq_{\mathrm{E}} \phi_i$, for all $i \in \underline{n}$.

[3] introduces a type inclusion relation '$\leq_{\mathrm{S}}$' for this system, as the least pre-order on $\mathcal{T}$ such that:

$$\begin{array}{c} \phi_1\cap\cdots\cap\phi_n \ \leq_{\mathrm{S}}\ \phi_i, \ \text{ for all } i \in \underline{n} \\ \tau \leq_{\mathrm{S}} \phi_i, \text{ for all } i \in \underline{n} \ \Rightarrow\ \tau \leq_{\mathrm{S}} \phi_1\cap\cdots\cap\phi_n \end{array}$$

Notice that this relation expresses the just the commutative and associative behaviour of intersection, and excludes the contra-variance. The following rule is admissible:

$$\frac{\Gamma \vdash M:\sigma}{\Gamma \vdash M:\tau}\ (\sigma \leq_{\mathrm{S}} \tau)$$

In fact, replacing rule $(\cap E)$ by the rule:

$$(\leq_{\mathrm{S}}) : \frac{}{\Gamma, x{:}\sigma \vdash x:\tau}\ (\sigma \leq_{\mathrm{S}} \tau)$$

would give exactly the same system in terms of derivable judgements; notice that rule $(\leq_{\mathrm{S}})$ combines $(\cap E)$ and $(\cap I)$. As for the difference in derivable statements, it is possible to derive $\vdash_{\mathrm{E}} \lambda x.x : (\alpha{\rightarrow}\phi){\rightarrow}(\alpha\cap\gamma){\rightarrow}\phi$, which is not possible in '$\vdash_{\mathrm{S}}$' (see Example 4.8).

[10] shows the approximation result

$$\Gamma \vdash_{\mathrm{S}} M:\sigma \iff \exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\mathrm{S}} A:\sigma],$$

with which, as in Section 6.2, the result

If $\Gamma \vdash_{\mathrm{BCD}} M:\sigma$, then there are $\Gamma',\sigma' \in \mathcal{T}$ such that $\Gamma' \vdash_{\mathrm{S}} M:\sigma'$, $\sigma' \leq \sigma$ and $\Gamma \leq \Gamma'$.

can be shown (cf. Theorem 6.13), with '$\leq$' the BCD-type inclusion relation.

Using the '$\leq_{\mathrm{S}}$', a strict filter model $\mathcal{F}_{\mathrm{S}}$ is constructed which corresponds to [33]'s model $\mathcal{D}_A$. Using this model, in [3] it is shown that, for all terms M, valuations ξ, $[\![M]\!]_\xi = \{\sigma \in \mathcal{T} \mid \Gamma_\xi \vdash_{\mathrm{S}} M:\sigma\}$; since $\mathcal{F}_{\mathrm{S}}$ is a λ-model, this implies '$\vdash_{\mathrm{S}}$' is closed for β-equality. Then [3] shows that strict type assignment is sound and complete with respect to inference type semantics: $\Gamma \vdash_{\mathrm{S}} M:\sigma \iff \Gamma \models M:\sigma$ (see also Section 7.3).

5.1 Principal pairs for '$\vdash_{\mathrm{S}}$'

The proof of the principal type property for '$\vdash_{\mathrm{S}}$' as presented in [6] is achieved in a way similar to, but significantly different from, the two techniques sketched above.

In that paper, three operations on pairs of context and types are defined: *substitution*, *expansion*, and *lifting*. The operation of lifting resembles the operation of rise as defined in [55], the operation of substitution is a modification of the one normally used, and the operation of expansion is a limited version of the one given in [21, 55].

In order to prove that the operations defined are sufficient, three subsets of the set of all pairs of context and type are defined, namely: principal pairs, ground pairs, and primitive pairs. (The definition of ground pairs coincides with the one given in [21].) 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 $\langle\Gamma,\sigma\rangle$ is a suitable pair for a term A in $\lambda\bot$-normal form, and $\langle\Pi,\pi\rangle$ is the principal pair for A, then there are a sequence of operations of expansion $\overrightarrow{Ex_i}$, an operation of lifting L, and a substitution S, such that

$$\langle\Gamma,\sigma\rangle = S\,(L\,(\overrightarrow{Ex}\,(\langle\Pi,\pi\rangle))).$$

This result is then generalised to arbitrary λ-terms.

Because of technical reasons, substitution is in [6] defined as the replacement of a type-variable φ by a type $\alpha \in \mathcal{T}_{\mathrm{S}} \cup \{\omega\}$, so it can also replace type-variables by the type constant ω (this is not needed in Section 10, where we show the principal pair property for '$\vdash_{\mathrm{E}}$'). 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, 'normalises while substituting'.

The operation of expansion, as defined in [6], corresponds to the one given in [21] and is a simplified version of the one defined in [55]. A difference is that in those definitions subtypes are collected, whereas the definition of expansion in [6] (see Definition 10.12) collects type-variables.

The operation of lifting as defined in [6] (see Definition 10.23) is based on the relation '$\leq_{\mathrm{E}}$', in the same way as the operation of rise is based on '$\leq$'. As shown there, that operation is not sound on all pairs $\langle \Gamma,\sigma \rangle$, so the following does not hold:

$$\Gamma \vdash_{\mathrm{S}} M{:}\sigma \;\&\; \sigma \leq_{\mathrm{E}} \tau \Rightarrow \Gamma \vdash_{\mathrm{S}} M{:}\sigma.$$

As a counter example, take $x{:}\sigma{\rightarrow}\sigma \vdash_{\mathrm{S}} x{:}\sigma{\rightarrow}\sigma$. Notice that $\sigma{\rightarrow}\sigma \leq_{\mathrm{E}} \sigma{\cap}\tau{\rightarrow}\sigma$, but it is impossible to derive $x{:}\sigma{\rightarrow}\sigma \vdash_{\mathrm{S}} x{:}\sigma{\cap}\tau{\rightarrow}\sigma$. (In fact, as argued in [6], 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 that uses the relation '$\leq_{\mathrm{E}}$', corresponds to an η-reduction step (see Theorem 4.7), and '$\vdash_{\mathrm{S}}$' is not closed for η-reduction. Since strict type assignment is *not* closed for '$\leq_{\mathrm{E}}$', and the operation of lifting implicitly applies '$\leq_{\mathrm{E}}$' to a derivation, it is clear that a conflict arises.

However, in [6] 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 [21]. 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 [6] follows from:

- Every principal pair is a ground pair.
- For every expansion Ex, if $\langle \Gamma,\sigma \rangle$ is a ground pair for A and $Ex(\langle \Gamma,\sigma \rangle) = \langle \Gamma',\sigma' \rangle$, then $\langle \Gamma',\sigma' \rangle$ is a ground pair for A.
- Every ground pair is a primitive pair.
- Lifting is a sound operation on primitive pairs: for all $A \in \mathcal{A}$, liftings L: if $\langle \Gamma,\sigma \rangle$ is a primitive pair for A, then $L(\langle \Gamma,\sigma \rangle)$ is a primitive pair for A.
- Every primitive pair is a (normal) pair.
- Substitution is a sound operation: If $\Gamma \vdash_{\mathrm{S}} A{:}\sigma$, then for all substitutions S: if $S(\langle \Gamma,\sigma \rangle) = \langle \Gamma',\sigma' \rangle$, then $\Gamma' \vdash_{\mathrm{S}} A{:}\sigma'$.

Although lifting is not sound on all pairs, using the results mentioned above it is possible to prove that the three operations defined in [6] are sufficient (complete): for every pair $\langle \Gamma,\sigma \rangle$ and $A \in \mathcal{A}$, if $\Gamma \vdash_{\mathrm{S}} A{:}\sigma$, then there exists a number of expansions, at most one lifting, and at most one substitution, such that $\langle \Gamma,\sigma \rangle$ can be obtained from $pp(A)$ by performing these operations in sequence. As in [55], this result is then generalised to arbitrary λ-terms (see Property 3.17 and Theorem 10.26).

6 Approximation and head-normalisation results

In this section, we will prove a number of results for '$\vdash_{\mathrm{E}}$'. First we will prove the approximation theorem (cf. Property 3.16); from this result, the well-known characterisation of head-normalisation and normalisation of λ-terms using intersection types follows easily, i.e., all terms having a head-normal form are typeable in '$\vdash_{\mathrm{E}}$' (with a type not equivalent to ω), and all terms having a normal form are typeable with a context and type that do not contain ω at all.

In [55], the approximation result is obtained through a normalisation of derivations, where all $(\to I)$–$(\to E)$ 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 [21, 15]; it requires a rather difficult notion of length of a derivation to show that this process terminates.) In this section, this result will be proven using the reducibility technique [57].

With this result, we will show that '$\vdash_{\mathrm{BCD}}$' is conservative over '$\vdash_{\mathrm{E}}$' presented here, and prove that all terms having a head-normal form are typeable in '$\vdash_{\mathrm{E}}$' (with a type different from ω).

6.1 Approximation result

In this subsection, the approximation theorem '$\Gamma \vdash_{\mathrm{E}} M:\sigma \iff \exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\mathrm{E}} A:\sigma]$' will be proven. For reasons of readability, we will be abbreviate $\exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\mathrm{E}} A:\sigma]$ by $\mathit{Appr}\,(\Gamma,M,\sigma)$.

First we show that type assignment is upward closed for '$\sqsubseteq$' (see Definition 2.6).

Lemma 6.1 $\Gamma \vdash_{\mathrm{E}} M:\sigma \ \&\ M \sqsubseteq M' \Rightarrow \Gamma \vdash_{\mathrm{E}} M':\sigma$.

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

The following basic properties are needed further on.

Lemma 6.2 *i)* $\mathit{Appr}\,(\Gamma,xM_1\cdots M_n,\sigma\to\phi)\ \&\ \mathit{Appr}\,(\Gamma,N,\sigma) \Rightarrow \mathit{Appr}\,(\Gamma,xM_1\cdots M_nN,\phi)$.

ii) $\mathit{Appr}\,(\Gamma\cup\{z{:}\sigma\},Mz,\phi)\ \&\ z\notin \mathit{fv}\,(M) \Rightarrow \mathit{Appr}\,(\Gamma,M,\sigma\to\phi)$.

iii) $\mathit{Appr}\,(\Gamma,M[N/x]\vec{P},\sigma) \Rightarrow \mathit{Appr}\,(\Gamma,(\lambda x.M)N\vec{P},\sigma)$.

Proof: *i)* $A \in \mathcal{A}(xM_1\cdots M_n)\ \&\ \Gamma \vdash_{\mathrm{E}} A:\sigma\to\phi\ \&\ A' \in \mathcal{A}(N)\ \&\ \Gamma \vdash_{\mathrm{E}} A':\sigma$
$\Rightarrow$ (2.7(i) & $(\to E)$ & $A \neq \bot$) $AA' \in \mathcal{A}(xM_1\cdots M_nN)\ \&\ \Gamma \vdash_{\mathrm{E}} AA':\phi$.

ii) $A \in \mathcal{A}(Mz)\ \&\ \Gamma,z{:}\sigma \vdash_{\mathrm{E}} A:\phi\ \&\ z\notin \mathit{fv}\,(M) \Rightarrow$ (2.7(ii))

a) $A \equiv A'z\ \&\ z \notin \mathit{fv}\,(A')\ \&\ A' \in \mathcal{A}(M)\ \&\ \Gamma,z{:}\sigma \vdash_{\mathrm{E}} A'z:\phi \Rightarrow$ (4.6)
$A' \in \mathcal{A}(M)\ \&\ \Gamma \vdash_{\mathrm{E}} A':\sigma\to\phi$.

b) $\lambda z.A \in \mathcal{A}(M)\ \&\ \Gamma,z{:}\sigma \vdash_{\mathrm{E}} A:\phi \Rightarrow \lambda z.A \in \mathcal{A}(M)\ \&\ \Gamma \vdash_{\mathrm{E}} \lambda z.A:\sigma\to\phi$.

iii) Since $M[N/x]\vec{P} =_\beta (\lambda x.M)N\vec{P}$, the result follows by Lemma 2.7(iii). ■

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

Definition 6.3 (Computability predicate) The predicate $\mathit{Comp}\,(\Gamma,M,\rho)$ is inductively defined by:

$$\begin{array}{rcl}
\mathit{Comp}\,(\Gamma,M,\varphi) & \iff & \mathit{Appr}\,(\Gamma,M,\varphi) \\
\mathit{Comp}\,(\Gamma,M,\sigma\to\phi) & \iff & (\ \mathit{Comp}\,(\Gamma',N,\sigma) \Rightarrow \mathit{Comp}\,(\cap\{\Gamma,\Gamma'\},MN,\phi)\ ) \\
\mathit{Comp}\,(\Gamma,M,\phi_1\cap\cdots\cap\phi_n) & \iff & \forall i \in \underline{n}\ [\mathit{Comp}\,(\Gamma,M,\phi_i)]
\end{array}$$

Notice that $Comp(\Gamma, M, \omega)$ holds as special case of the third part.

We will now show that the computability predicate is closed for '$\leq_{\mathrm{E}}$', for which we first show:

Lemma 6.4 i) If $Comp(\Gamma, M, \sigma)$, and $\Gamma' \leq_{\mathrm{E}} \Gamma$, then $Comp(\Gamma', M, \sigma)$.
ii) If $Comp(\Gamma, M, \sigma)$, and $\sigma \leq_{\mathrm{E}} \tau$, then $Comp(\Gamma, M, \tau)$.

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

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

Lemma 6.5 $Comp(\Gamma, M[N/x]\vec{P}, \sigma) \Rightarrow Comp(\Gamma, (\lambda x.M)N\vec{P}, \sigma)$.

Proof: By induction on the definition of *Comp*.

$$\begin{array}{ll}(\sigma = \varphi): & Comp(\Gamma, M[N/x]\vec{P}, \varphi) \Rightarrow Appr(\Gamma, M[N/x]\vec{P}, \varphi) \Rightarrow (6.2(iii)) \\ & Appr(\Gamma, (\lambda x.M)N\vec{P}, \varphi) \Rightarrow Comp(\Gamma, (\lambda x.M)N\vec{P}, \varphi).\end{array}$$

$$\begin{array}{lll}(\sigma = \tau\to\phi): & Comp(\Gamma, M[N/x]\vec{P}, \tau\to\phi) & \Rightarrow (6.3) \\ & (Comp(\Gamma', Q, \tau) \Rightarrow Comp(\cap\{\Gamma, \Gamma'\}, M[N/x]\vec{P}Q, \phi)) & \Rightarrow (IH) \\ & (Comp(\Gamma', Q, \tau) \Rightarrow Comp(\cap\{\Gamma, \Gamma'\}, (\lambda x.M)N\vec{P}Q, \phi)) & \Rightarrow (6.3) \\ & Comp(\Gamma, (\lambda x.M)N\vec{P}, \tau\to\phi). & \end{array}$$

$(\sigma = \phi_1 \cap \cdots \cap \phi_n)$: By induction. ■

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

Theorem 6.6 *i) $Appr(\Gamma, xM_1 \cdots M_n, \rho) \Rightarrow Comp(\Gamma, xM_1 \cdots M_n, \rho)$.*
ii) $Comp(\Gamma, M, \rho) \Rightarrow Appr(\Gamma, M, \rho)$.

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

$$\begin{array}{lll} i) & Appr(\Gamma, xM_1 \cdots M_n, \sigma\to\phi) & \Rightarrow (IH(ii)) \\ & (Comp(\Gamma', N, \sigma) \Rightarrow Appr(\Gamma, xM_1 \cdots M_n, \sigma\to\phi) \;\&\; Appr(\Gamma', N, \sigma)) & \Rightarrow (6.2(i)) \\ & (Comp(\Gamma', N, \sigma) \Rightarrow Appr(\cap\{\Gamma, \Gamma'\}, xM_1 \cdots M_n N, \phi)) & \Rightarrow (IH(i)) \\ & (Comp(\Gamma', N, \sigma) \Rightarrow Comp(\cap\{\Gamma, \Gamma'\}, xM_1 \cdots M_n N, \phi)) & \Rightarrow (6.3) \\ & Comp(\Gamma, xM_1 \cdots M_n, \sigma\to\phi). & \end{array}$$

$$\begin{array}{lll} ii) & Comp(\Gamma, M, \sigma\to\phi) \;\&\; z \notin fv(M) & \Rightarrow (IH(i)) \\ & Comp(\Gamma, M, \sigma\to\phi) \;\&\; Comp(\{z{:}\sigma\}, z, \sigma) \;\&\; z \notin fv(M) & \Rightarrow (6.3) \\ & Comp(\cap\{\Gamma, \{z{:}\sigma\}\}, Mz, \phi) \;\&\; z \notin fv(M) & \Rightarrow (IH(ii)) \\ & Appr(\cap\{\Gamma, \{z{:}\sigma\}\}, Mz, \phi) \;\&\; z \notin fv(M) & \Rightarrow (6.2(ii)) \\ & Appr(\Gamma, M, \sigma\to\phi). \;\blacksquare & \end{array}$$

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

Corollary 6.7 $Comp(\{x{:}\sigma\}, x, \sigma)$, for all x, σ.

We now come to the main result of this section, which states that a computable extension of a typeable term yields a computable object.

Theorem 6.8 (Replacement Theorem) *If $\{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\} \vdash_{\mathrm{E}} M : \sigma$, and, for every $i \in \underline{n}$, $Comp(\Gamma_i, N_i, \mu_i)$, then $Comp(\cap_n \Gamma_i, M[\overrightarrow{N_i/x_i}], \sigma)$.*

Proof: By induction on the structure of derivations; let $\{x_1{:}\mu_1,\ldots,x_n{:}\mu_n\} = \Gamma_0$, and $\Gamma' = \cap_n \Gamma_i$.

(Ax): Then $M \equiv x_j$, for some $j \in \underline{n}$, $\mu_j \leq_{\text{E}} \sigma$, and $M[\overrightarrow{N_i/x_i}] \equiv x_j[\overrightarrow{N_i/x_i}] \equiv N_j$.

$$\begin{array}{ll} \mathit{Comp}\,(\Gamma_j,N_j,\mu_j) & \Rightarrow (\mu_j \leq_{\text{E}} \sigma \ \&\ 6.4) \\ \mathit{Comp}\,(\Gamma_j,N_j,\sigma) & \Rightarrow (\Gamma' \leq_{\text{E}} \Gamma_j \ \&\ 6.4) \\ \mathit{Comp}\,(\Gamma',N_j,\sigma). & \end{array}$$

$(\to I)$: Then $M \equiv \lambda y.M'$, $\sigma = \rho\to\phi$, and $\Gamma_0,y{:}\rho \vdash_{\text{E}} M'{:}\phi$.

$$\begin{array}{ll} \forall i \in \underline{n}\ [\mathit{Comp}\,(\Gamma_i,N_i,\mu_i)]\ \&\ \Gamma_0,y{:}\rho \vdash_{\text{E}} M'{:}\phi & \Rightarrow (IH\ \&\ 6.4(i)) \\ (\mathit{Comp}\,(\Gamma',N,\rho) \Rightarrow \mathit{Comp}\,(\cap\{\Gamma',\Gamma'\},M'[\overrightarrow{N_i/x_i},N/y],\phi)) & \Rightarrow (6.5) \\ (\mathit{Comp}\,(\Gamma',N,\rho) \Rightarrow \mathit{Comp}\,(\cap\{\Gamma',\Gamma'\},(\lambda y.M'[\overrightarrow{N_i/x_i}])N,\phi)) & \Rightarrow (6.3) \\ \mathit{Comp}\,(\Gamma',(\lambda y.M')[\overrightarrow{N_i/x_i}],\rho\to\phi). & \end{array}$$

$(\to E)$: Then $M \equiv M_1M_2$, $\Gamma_0 \vdash_{\text{E}} M_1{:}\rho\to\sigma$, and $\Gamma_0 \vdash_{\text{E}} M_2{:}\rho$.

$$\begin{array}{ll} \forall i \in \underline{n}\ [\mathit{Comp}\,(\Gamma_i,N_i,\mu_i)]\ \&\ \Gamma_0 \vdash_{\text{E}} M_1{:}\rho\to\sigma\ \&\ \Gamma_0 \vdash_{\text{E}} M_2{:}\rho & \Rightarrow (IH) \\ \mathit{Comp}\,(\Gamma',M_1[\overrightarrow{N_i/x_i}],\rho\to\sigma)\ \&\ \mathit{Comp}\,(\Gamma',M_2[\overrightarrow{N_i/x_i}],\rho) & \Rightarrow (6.3) \\ \mathit{Comp}\,(\Gamma',(M_1M_2)[\overrightarrow{N_i/x_i}],\sigma). & \end{array}$$

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

We can now show the approximation result.

Theorem 6.9 (Approximation theorem) $\Gamma \vdash_{\text{E}} M{:}\sigma \iff \exists\, A \in \mathcal{A}(M)\ [\Gamma \vdash_{\text{E}} A{:}\sigma]$.

Proof: $(\Rightarrow)$:

$$\begin{array}{ll} \Gamma \vdash_{\text{E}} M{:}\sigma & \Rightarrow (6.8\ \&\ 6.7) \\ \mathit{Comp}\,(\Gamma,M,\sigma) & \Rightarrow (6.6(ii)) \\ \exists\, A \in \mathcal{A}(M)\ [\Gamma \vdash_{\text{E}} A{:}\sigma]. & \end{array}$$

$(\Leftarrow)$: Let $A \in \mathcal{A}(M)$ be such that $\Gamma \vdash_{\text{E}} A{:}\sigma$. Since $A \in \mathcal{A}(M)$, there is an M' such that $M' =_\beta M$ and $A \sqsubseteq M'$. Then, by Lemma 6.1, $\Gamma \vdash_{\text{E}} M'{:}\sigma$ and, by Theorem 4.10, also $\Gamma \vdash_{\text{E}} M{:}\sigma$. ■

6.2 The relation between '$\vdash_{\text{BCD}}$' and '$\vdash_{\text{E}}$'

We first show that '$\vdash_{\text{E}}$' is the nucleus of '$\vdash_{\text{BCD}}$' (see Section 3.5): we will show that, for any derivation in '$\vdash_{\text{BCD}}$', it is possible to find an equivalent derivation in '$\vdash_{\text{E}}$'.

The proof is based on the fact that for every $\sigma \in \mathcal{T}_{\text{BCD}}$ there is a $\sigma^* \in \mathcal{T}$ (the normalised version of σ) such that $\sigma \sim_{\text{E}} \sigma^*$, and the Approximation Theorem 6.9.

Definition 6.10 ((cf. [37])) *i)* For every $\sigma \in \mathcal{T}_{\text{BCD}}$, $\sigma^* \in \mathcal{T}$ is inductively defined as follows:

$$\begin{array}{rcll} \varphi^* & = & \varphi & \\ (\sigma\to\tau)^* & = & \cap_n(\sigma^*\to\tau_i), & \text{if } \tau^* = \cap_n\tau_i\ (n \geq 0) \\ (\cap_n\tau_i)^* & = & \cap_m\sigma_i, & \text{where } \{\sigma_1,\ldots,\sigma_m\} = \{\tau_i^* \in \{\tau_1^*,\ldots,\tau_n^*\} \mid \tau_i^* \neq \omega\} \end{array}$$

ii) '$\cdot^*$' is extended to contexts by: $\Gamma^* = \{x{:}\sigma^* \mid x{:}\sigma \in \Gamma\}$.

Notice that $\omega^* = \omega$ as special case of the third part, and that $(\sigma\to\omega)^* = \omega$, as special case of the second. Since $\mathcal{T}$ is a proper subset of $\mathcal{T}_{\text{BCD}}$, σ^* is also defined for $\sigma \in \mathcal{T}$.

Lemma 6.11 i) (c.f. [37]) For every $\sigma \in \mathcal{T}_{\text{BCD}}$, $\sigma \sim \sigma^*$.

ii) $\mathcal{T}_{\text{BCD}}/\sim$ is isomorphic to $\mathcal{T}/\sim_{\text{E}}$.

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

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

Proof: Easy. ■

As in Section 10, the proof for the main theorem of this section is achieved by proving first that, for every term in $\mathcal{A}$ typeable in '$\vdash_{\text{BCD}}$', a derivation in '$\vdash_{\text{E}}$' can be built for which context and type in the conclusion are equivalent, and afterwards generalising this result to arbitrary λ-terms. This depends on the approximation theorem for '$\vdash_{\text{BCD}}$' (Property 3.16) and Theorem 6.9.

We can directly link '$\vdash_{\text{BCD}}$' and '$\vdash_{\text{E}}$' on approximants via normalised types as follows:

Theorem 6.12 *$\Gamma \vdash_{\text{BCD}} A:\sigma \Rightarrow \Gamma^* \vdash_{\text{E}} A:\sigma^*$.*

Proof: By easy induction on the structure of terms in $\mathcal{A}$, using Lemma 6.11(*iii*). ■

The relation between the two different notions of type assignment on normal terms can now be formulated as follows:

Theorem 6.13 *$\Gamma \vdash_{\text{BCD}} M:\sigma \Rightarrow \Gamma^* \vdash_{\text{E}} M:\sigma^*$.*

Proof:
$$\begin{array}{ll} \Gamma \vdash_{\text{BCD}} M:\sigma & \Rightarrow (3.16) \\ \exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\text{BCD}} A:\sigma] & \Rightarrow (6.12) \\ \exists A \in \mathcal{A}(M)\ [\Gamma^* \vdash_{\text{E}} A:\sigma^*] & \Rightarrow (6.9) \\ \Gamma^* \vdash_{\text{E}} M:\sigma^*. \ ■ & \end{array}$$

So, for every derivable judgement in '$\vdash_{\text{BCD}}$' there exists an equivalent judgement in '$\vdash_{\text{E}}$'.

The last result allows us to show that '$\vdash_{\text{BCD}}$' is a conservative extension of '$\vdash_{\text{E}}$'.

Theorem 6.14 (Conservativity) *Let Γ just contain types in $\mathcal{T}$, and $\sigma \in \mathcal{T}$, then: if $\Gamma \vdash_{\text{BCD}} M:\sigma$, then $\Gamma \vdash_{\text{E}} M:\sigma$.*

Proof: $\Gamma \vdash_{\text{BCD}} M:\sigma \Rightarrow (6.13)\ \Gamma^* \vdash_{\text{E}} M:\sigma^* \Rightarrow (6.11(iv))\ \Gamma \vdash_{\text{E}} M:\sigma$. ■

Obviously, since '$\vdash_{\text{E}}$' is a subsystem of '$\vdash_{\text{BCD}}$', the implication in the other direction also holds: if $\Gamma \vdash_{\text{E}} M:\sigma$, then $\Gamma \vdash_{\text{BCD}} M:\sigma$.

6.3 Characterisation of (head) normalisation

Using the approximation result, the following head-normalisation result becomes easy to show.

Theorem 6.15 (Head-normalisation) *$\exists \Gamma,\phi\ [\Gamma \vdash_{\text{E}} M:\phi] \Longleftrightarrow M$ has a head-normal form.*

Proof: ($\Rightarrow$): If $\Gamma \vdash_{\text{E}} M:\phi$, then, by Theorem 6.9, there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\text{E}} A:\phi$. By Definition 2.5, there exists $M' =_\beta M$ such that $A \sqsubseteq M$. Since $\phi \in \mathcal{T}_{\text{s}}$, $A \not\equiv \bot$, 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 $\lambda x.M_1$ or $xM_1 \cdots M_n$, with $M_i \in \Lambda$, for all $i \in \underline{n}$. Then either:

a) $M' \equiv \lambda x.M_1$. Since M_1 is in head-normal form, by induction there are Γ,ϕ such that $\Gamma \vdash_{\text{E}} M_1:\phi$. If $x{:}\tau \in \Gamma$, then $\Gamma \backslash x \vdash_{\text{E}} \lambda x.M_1:\tau \rightarrow \phi$; otherwise $\Gamma \vdash_{\text{E}} \lambda x.M_1:\omega \rightarrow \phi$.

b) $M' \equiv xM_1 \cdots M_n, (n \geq 0)$. Then $\{x{:}\omega \rightarrow \cdots \rightarrow \omega \rightarrow \phi\} \vdash_{\text{E}} xM_1 \cdots M_n:\phi$.

So there exists Γ,ϕ such that $\Gamma \vdash_{\text{E}} M':\phi$, and, by Theorem 4.10, we get $\Gamma \vdash_{\text{E}} M:\phi$. ■

To prepare the characterisation of normalisability by assignable types, first we prove 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 normalisable terms are typeable without ω.

Lemma 6.16 i) If $\Gamma \vdash_{\text{E}} A:\sigma$ and Γ,σ are ω-free, then A is $\bot$-free.

ii) If A is $\bot$-free, then there are ω-free Γ and σ, such that $\Gamma \vdash_{\text{E}} A:\sigma$.

Proof: By induction on A.

i) As before, only the part $\sigma = \phi \in \mathcal{T}_{\rm s}$ is shown.

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

$(A \equiv \lambda x.A')$: Then $\phi = \rho{\to}\psi$, and $\Gamma, x{:}\rho \vdash_{\rm E} A : \psi$. Since Γ, ϕ are ω-free, so are $\Gamma, x{:}\rho$ and ψ, so, by induction, A' is $\bot$-free, so also $\lambda x.A'$ is $\bot$-free.

$(A \equiv xA_1 \cdots A_n)$: Then, by $({\to}E)$ and (Ax), there are σ_i, τ_i $(i \in \underline{n}), \psi$, such that $\Gamma \vdash_{\rm E} A_i : \phi_i$ for all $i \in \underline{n}$, $x{:}\tau_1{\to}\cdots{\to}\tau_n{\to}\psi \in \Gamma$, and $\tau_1{\to}\cdots{\to}\tau_n{\to}\psi \leq_{\rm E} \sigma_1{\to}\cdots{\to}\sigma_n{\to}\psi$. So, especially, for every $i \in \underline{n}$, $\sigma_i \leq_{\rm E} \tau_i$. By Theorem 4.7, also $\Gamma \vdash_{\rm E} A_i : \tau_i$, for every $i \in \underline{n}$. Since each τ_i occurs in Γ, all are ω-free, so by induction each A_i is $\bot$-free. Then also $xA_1 \cdots A_n$ is $\bot$-free.

ii) $(A \equiv \lambda x.A')$: By induction there are Γ, ψ such that $\Gamma \vdash_{\rm E} A' : \psi$ and Γ, ψ are ω-free. If x does not occur in Γ, take an ω-free $\sigma \in \mathcal{T}$. Otherwise, there exist $x{:}\tau \in \Gamma$, and τ is ω-free. In any case, $\Gamma \backslash x \vdash_{\rm E} \lambda x.A' : \tau{\to}\psi$, and $\Gamma \backslash x$ and $\tau{\to}\psi$ are ω-free.

$(A \equiv xA_1 \cdots A_n$, *with* $(n \geq 0))$: By induction there are Γ_i, σ_i $(i \in \underline{n})$ such that, for every $i \in \underline{n}$, $\Gamma_i \vdash_{\rm E} A_i : \sigma_i$, and Γ_i, σ_i are ω-free. Take any ϕ strict, such that ω does not occur in ϕ, and $\Gamma = \cap_n \Gamma_i \cap \{x{:}\sigma_1{\to}\cdots{\to}\sigma_n{\to}\phi\}$. Then $\Gamma \vdash_{\rm E} xA_1 \cdots A_n : \phi$, and Γ and ϕ are ω-free. ■

Notice that, by construction, in part (*ii*), the type constant ω is not used at all in the derivation, a fact we need in the proof for Lemma 8.3, in support of the strong normalisation result, Theorem 8.7.

Now, as shown in [3] for '$\vdash_{\rm S}$' and in [15] for '$\vdash_{\rm BCD}$', it is possible to prove that we can characterise normalisation.

Theorem 6.17 (Normalisation) $\exists \Gamma, \sigma\ [\Gamma \vdash_{\rm E} M : \sigma\ \&\ \Gamma, \sigma$ *ω-free*$] \iff M$ *has a normal form.*

Proof: $(\Rightarrow)$: If $\Gamma \vdash_{\rm E} M : \sigma$, then, by Theorem 6.9, there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\rm E} A : \sigma$. Then, by Lemma 6.16(*i*), this A is $\bot$-free. By Definition 2.5, there exists $M' =_\beta M$ such that $A \sqsubseteq M'$. Since A is $\bot$-free, we have $A \equiv M'$, 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 6.16(*ii*), there are ω-free Γ, σ such that $\Gamma \vdash_{\rm E} M' : \sigma$, and, by Theorem 4.10, $\Gamma \vdash_{\rm E} M : \sigma$. ■

As for the fact that typeability *per se* does not guarantee termination, consider the following.

Example 6.18 Take $\Theta = \lambda xy.y(xxy)$, then $\Theta\Theta$ (Turing's fixed-point combinator) is typeable. First we derive $\mathcal{D}_1$ (where $\Gamma = \{x{:}(\alpha{\to}\beta{\to}\gamma)\cap\alpha, y{:}(\gamma{\to}\delta)\cap\beta\}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \gamma{\to}\delta} \quad \dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : \alpha{\to}\beta{\to}\gamma} \quad \dfrac{}{\Gamma \vdash x : \alpha}}{\Gamma \vdash xx : \beta{\to}\gamma}\,({\to}E) \quad \dfrac{}{\Gamma \vdash y : \beta}}{\Gamma \vdash xxy : \gamma}\,({\to}E)}{\Gamma \vdash y(xxy) : \delta}\,({\to}E)}{B\backslash y \vdash \lambda y.y(xxy) : ((\gamma{\to}\delta)\cap\beta){\to}\delta}\,({\to}I)}{\vdash \Theta : ((\alpha{\to}\beta{\to}\gamma)\cap\alpha){\to}((\gamma{\to}\delta)\cap\beta){\to}\delta}\,({\to}I)$$

Let $\tau = ((\alpha{\to}\beta{\to}\gamma)\cap\alpha){\to}((\gamma{\to}\delta)\cap\beta){\to}\delta$ (i.e. the type derived in $\mathcal{D}_1$), then we can derive $\mathcal{D}_2$:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x{:}\tau, y{:}\omega\to\phi \vdash y : \omega\to\phi} \quad \dfrac{}{x{:}\tau, y{:}\omega\to\phi \vdash xxy : \omega}\,(\cap I)}{x{:}\tau, y{:}\omega\to\phi \vdash y(xxy) : \phi}\,(\to E)}{x{:}\tau \vdash \lambda y.y(xxy) : (\omega\to\phi)\to\phi}\,(\to I)}{\vdash \Theta : \tau\to(\omega\to\phi)\to\phi}\,(\to I)$$

Notice that, in fact, the type τ is irrelevant here; since x occurs only in a subterm that is typed with ω, any type for x could be used here. By rule $(\to E)$ we can now construct:

$$\dfrac{\dfrac{\boxed{\quad\mathcal{D}_2\quad}}{\vdash \Theta : \tau\to(\omega\to\phi)\to\phi} \qquad \dfrac{\boxed{\mathcal{D}_1}}{\vdash \Theta : \tau}}{\vdash \Theta\Theta : (\omega\to\phi)\to\phi}\,(\to E)$$

Notice that this term is not strongly normalisable, since

$$\Theta\Theta \to_\beta \lambda y.(\Theta\Theta y) \twoheadrightarrow_\beta \lambda y.(y(\Theta\Theta y)) \twoheadrightarrow_\beta \cdots.$$

so typeability does not enforce termination.

The problem is more subtle than that, however. One of the roles of ω in the proofs above is to cover terms that will disappear during reduction. In view of the normalisation results above, a natural thought is to assume that, when not allowing a redex to occur in a subterm typed with ω, this would ensure termination. This is not the case.

Example 6.19 Notice that, in the derivation $\mathcal{D} :: \vdash_{\text{E}} \Theta\Theta : (\omega\to\phi)\to\phi$ constructed above, there occurs only one redex in $\Theta\Theta$, and that this redex is not typed with ω. We can now type $\lambda y.(\Theta\Theta y)$ as follows:

$$\dfrac{\dfrac{\dfrac{\boxed{\quad\mathcal{D}\quad}}{y{:}\omega\to\phi \vdash \Theta\Theta : (\omega\to\phi)\to\phi} \qquad \dfrac{}{y{:}\omega\to\phi \vdash y : \omega\to\phi}}{y{:}\omega\to\phi \vdash \Theta\Theta y : \phi}}{\vdash \lambda y.(\Theta\Theta y) : (\omega\to\phi)\to\phi}$$

Again, the redex is not covered with ω. We can even derive:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \phi\to\psi} \qquad \dfrac{\dfrac{\boxed{\quad\mathcal{D}\quad}}{\Gamma \vdash \Theta\Theta : (\omega\to\phi)\to\phi} \qquad \dfrac{}{\Gamma \vdash y : \omega\to\phi}}{\Gamma \vdash \Theta\Theta y : \phi}}{\Gamma \vdash y(\Theta\Theta y) : \psi}}{\vdash \lambda y.(y(\Theta\Theta y)) : (\phi\to\psi)\cap(\omega\to\phi)\to\psi}$$

(where $\Gamma = y{:}(\phi\to\psi)\cap(\omega\to\phi)$) and again, the redex is not covered with ω; we can repeat this construction, in fact, for all the reducts of $\Theta\Theta$.

We will return to this example in Section 9.

7 Semantics and completeness for essential type assignment

As was shown in [15] for '$\vdash_{\text{BCD}}$' (see Property 3.15), in this section we will show that '$\vdash_{\text{E}}$' is sound and complete with respect to the simple type semantics; a similar result was shown in [3] for '$\vdash_{\text{S}}$', albeit with respect to inference type semantics.

7.1 The Essential Filter model

As in [15] (see Definition 3.12), a filter λ-model can be constructed, where terms will be interpreted by their assignable types. First we define filters as sets of types closed for intersection and upward closed for '$\leq_{\textsc{e}}$'.

Definition 7.1 (Essential Filters) *i)* A subset d of $\mathcal{T}$ is an *essential filter* if and only if:

$$\begin{array}{rcl} \phi_i \in d \ (\forall i \in \underline{n}, n \geq 0) & \Rightarrow & \phi_1 \cap \cdots \cap \phi_n \in d \\ \tau \in d \ \& \ \tau \leq_{\textsc{e}} \sigma & \Rightarrow & \sigma \in d \end{array}$$

ii) If V is a subset of $\mathcal{T}$, then $\uparrow_{\textsc{e}} V$, the *essential filter generated by* V, is the smallest essential filter that contains V, and $\uparrow_{\textsc{e}} \sigma = \uparrow_{\textsc{e}} \{\sigma\}$.

iii) The domain $\mathcal{F}_{\textsc{e}}$ is defined by: $\mathcal{F}_{\textsc{e}} = \{ d \subseteq \mathcal{T} \mid d \text{ is an essential filter} \}$. Application on $\mathcal{F}_{\textsc{e}}$ is defined by:

$$d \cdot e = \uparrow_{\textsc{e}} \{ \phi \mid \exists \sigma \in e \ [\sigma \rightarrow \phi \in d] \}.$$

Notice that this definition is much the same as Definition 3.12, with the exception of the last part, where the above definition *forces* the creation of a filter. Contrary to the case for '$\vdash_{\textsc{bcd}}$', application *must* be forced to yield filters, since in each arrow type scheme $\sigma \rightarrow \phi \in \mathcal{T}$, ϕ is strict, and filters need to be closed for intersection.

$\langle \mathcal{F}_{\textsc{e}}, \subseteq \rangle$ is a cpo and henceforward it will be considered with the corresponding Scott topology. Notice that, as a BCD filter, an essential filter is never empty since, for all d, $\omega \in d$ by the first clause of Definition 7.1(*i*), and that an essential filter is a BCD filter, but not vice-versa.

For essential filters the following properties hold:

Lemma 7.2 *i)* $\sigma \in \uparrow_{\textsc{e}} \tau \iff \tau \leq_{\textsc{e}} \sigma$.

ii) $\sigma \in \uparrow_{\textsc{e}} \{ \phi \mid \Gamma \vdash_{\textsc{e}} M : \phi \} \iff \Gamma \vdash_{\textsc{e}} M : \sigma$.

Proof: Easy. ■

In particular, by part (*ii*), $\{ \sigma \mid \Gamma \vdash_{\textsc{e}} M : \sigma \} \in \mathcal{F}_{\textsc{e}}$.

Definition 7.3 The domain constructors $F : \mathcal{F}_{\textsc{e}} \rightarrow [\mathcal{F}_{\textsc{e}} \rightarrow \mathcal{F}_{\textsc{e}}]$ and $G : [\mathcal{F}_{\textsc{e}} \rightarrow \mathcal{F}_{\textsc{e}}] \rightarrow \mathcal{F}_{\textsc{e}}$ are defined by:

$$\begin{array}{rcl} F \ d \ e & = & d \cdot e \\ G \ f & = & \uparrow_{\textsc{e}} \{ \sigma \rightarrow \phi \mid \phi \in f(\uparrow_{\textsc{e}} \sigma) \} \end{array}$$

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

Theorem 7.4 (Filter model) *$\langle \mathcal{F}_{\textsc{e}}, \cdot, F, G \rangle$ is a λ-model.*

Proof: By [14].5.4.1 it is sufficient to prove that $F \circ G = Id_{[\mathcal{F}_{\textsc{e}} \rightarrow \mathcal{F}_{\textsc{e}}]}$.

$$\begin{array}{rll} F \circ G \ f \ d = & F \ (G \ f) \ d = F \ (\uparrow_{\textsc{e}} \{ \sigma \rightarrow \phi \mid \phi \in f(\uparrow_{\textsc{e}} \sigma) \}) \ d & = \\ & \uparrow_{\textsc{e}} \{ \psi \mid \exists \rho \in d \ [\rho \rightarrow \psi \in \uparrow_{\textsc{e}} \{ \sigma \rightarrow \phi \mid \phi \in f(\uparrow_{\textsc{e}} \sigma) \}] \} & = \ (7.2(i)) \\ & \uparrow_{\textsc{e}} \{ \psi \mid \exists \rho \in d \ [\psi \in f(\uparrow_{\textsc{e}} \rho)] \} & = f(d). \ ■ \end{array}$$

Definition 7.5 (Term interpretation) *i)* Let $\mathcal{M}$ be a λ-model, and ξ be a valuation of term-variables in $\mathcal{M}$; $[\![\cdot]\!]^{\mathcal{M}}_{\xi}$, the interpretation of terms in $\mathcal{M}$ via ξ is inductively defined by:

$$\begin{array}{rcl} [\![x]\!]^{\mathcal{M}}_{\xi} & = & \xi(x) \\ [\![MN]\!]^{\mathcal{M}}_{\xi} & = & F [\![M]\!]^{\mathcal{M}}_{\xi} [\![N]\!]^{\mathcal{M}}_{\xi} \\ [\![\lambda x.M]\!]^{\mathcal{M}}_{\xi} & = & G(\boldsymbol{\lambda} d \in \mathcal{M}. [\![M]\!]^{\mathcal{M}}_{\xi(d/x)}) \end{array}$$

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

Since $\mathcal{F}_{\mathrm{E}}$ is the model studied here, $[\![\cdot]\!]_\xi$ will stand for $[\![\cdot]\!]_\xi^{\mathcal{F}_E}$. Notice that Γ_ξ is not really a context, since it can contain infinitely many statements with subject x; however, for all design and purposes, it can be regarded as one, accepting the concept of an infinite intersection.

Theorem 7.6 *For all M,ξ: $[\![M]\!]_\xi = \{\sigma \mid \Gamma_\xi \vdash_{\mathrm{E}} M{:}\sigma\}$.*

Proof: By induction on the structure of λ-terms.

i) $[\![x]\!]_\xi = \xi(x)$. If $\sigma \in \xi(x)$, then certainly $\Gamma_\xi \vdash_{\mathrm{E}} x{:}\sigma$. Assume $\Gamma_\xi \vdash_{\mathrm{E}} x{:}\sigma$; if $x{:}\rho \in \Gamma_\xi$, then $\rho \leq_{\mathrm{E}} \sigma$, so $\sigma \in {\uparrow_{\mathrm{E}}}\rho$. Since $\rho \in \xi(x)$, also ${\uparrow_{\mathrm{E}}}\rho \subseteq \xi(x)$, so $\sigma \in \xi(x)$.

ii)
$$\begin{array}{ll}
[\![MN]\!]_\xi = F\ [\![M]\!]_\xi\ [\![N]\!]_\xi = [\![M]\!]_\xi \cdot [\![N]\!]_\xi & = (IH) \\
\{\rho \mid \Gamma_\xi \vdash_{\mathrm{E}} M{:}\rho\} \cdot \{\rho \mid \Gamma_\xi \vdash_{\mathrm{E}} N{:}\rho\} & = (7.1(iii)) \\
{\uparrow_{\mathrm{E}}}\{\phi \mid \exists \sigma \in \{\rho \mid \Gamma_\xi \vdash_{\mathrm{E}} N{:}\rho\}\ [\sigma{\to}\phi \in \{\rho \mid \Gamma_\xi \vdash_{\mathrm{E}} M{:}\rho\}]\} & = \\
{\uparrow_{\mathrm{E}}}\{\phi \mid \exists \sigma\ [\Gamma_\xi \vdash_{\mathrm{E}} N{:}\sigma\ \&\ \Gamma_\xi \vdash_{\mathrm{E}} M{:}\sigma{\to}\phi]\} & = ({\to}E) \\
{\uparrow_{\mathrm{E}}}\{\phi \mid \Gamma_\xi \vdash_{\mathrm{E}} MN{:}\phi\} & = (7.2(ii)) \\
\{\sigma \mid \Gamma_\xi \vdash_{\mathrm{E}} MN{:}\sigma\} &
\end{array}$$

iii)
$$\begin{array}{ll}
[\![\lambda x.M]\!]_\xi = G(\lambda\!\!\!\lambda d \in \mathcal{F}_{\mathrm{E}}.[\![M]\!]_{\xi(d/x)}) & = (IH) \\
G(\lambda\!\!\!\lambda d \in \mathcal{F}_{\mathrm{E}}.\{\rho \mid \Gamma_{\xi(d/x)} \vdash_{\mathrm{E}} M{:}\rho\}) & = \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \phi \in (\lambda\!\!\!\lambda d \in \mathcal{F}_{\mathrm{E}}.\{\rho \mid \Gamma_{\xi(d/x)} \vdash_{\mathrm{E}} M{:}\rho\})({\uparrow_{\mathrm{E}}}\tau)\} & = \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \phi \in \{\rho \mid \Gamma_{\xi({\uparrow_{\mathrm{E}}}\tau/x)} \vdash_{\mathrm{E}} M{:}\rho\}\} & = \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \Gamma_{\xi({\uparrow_{\mathrm{E}}}\tau/x)} \vdash_{\mathrm{E}} M{:}\phi\} & = \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \Gamma_\xi \backslash x \cup \{x{:}\psi \mid \psi \in {\uparrow_{\mathrm{E}}}\tau\} \vdash_{\mathrm{E}} M{:}\phi\} & = (7.2(i)\ \&\ 4.5) \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \Gamma_\xi \backslash x, x{:}\tau \vdash_{\mathrm{E}} M{:}\phi\} & = ({\to}I) \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \Gamma_\xi \backslash x \vdash_{\mathrm{E}} \lambda x.M{:}\tau{\to}\phi\} & = (4.4) \\
{\uparrow_{\mathrm{E}}}\{\tau{\to}\phi \mid \Gamma_\xi \vdash_{\mathrm{E}} \lambda x.M{:}\tau{\to}\phi\} & = (({\to}I)\ \&\ 7.2(ii)) \\
\{\sigma \mid \Gamma_\xi \vdash_{\mathrm{E}} \lambda x.M{:}\sigma\}.\ \blacksquare &
\end{array}$$

7.2 Soundness and completeness of essential type assignment

We will now come to the proof for completeness of type assignment, as was also shown for '$\vdash_{\mathrm{BCD}}$' in [15]. A completeness result was also shown in [3], but for '$\vdash_{\mathrm{S}}$', and with respect to an inference type semantics; the completeness result we show here is, as in [15], achieved with respect to a simple type semantics.

The method followed in [15] for the proof of completeness of type assignment is to define a type interpretation v that satisfies: for all types σ, $v(\sigma) = \{d \in \mathcal{F}_\cap \mid \sigma \in d\}$. The approach taken here is to define a function, and to show that it is a simple type interpretation.

Definition 7.7 *i)* ν_0 is defined by: $\nu_0(\sigma) = \{d \in \mathcal{F}_{\mathrm{E}} \mid \sigma \in d\}$.

ii) $\xi_\Gamma(x) = \{\sigma \in \mathcal{T} \mid \Gamma \vdash_{\mathrm{E}} x{:}\sigma\} = {\uparrow_{\mathrm{E}}}\tau$, where $x{:}\tau \in \Gamma$.

Lemma 7.8 *$\sigma \leq_{\mathrm{E}} \tau$ implies $\nu_0(\sigma) \subseteq \nu_0(\tau)$.*

Proof: Easy. ■

Theorem 7.9 *The map ν_0 is a simple type interpretation.*

Proof: It is sufficient to check the conditions of Definition 3.13:

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

$$\begin{array}{ll}
\forall e\,[e \in \nu_0(\sigma) \Rightarrow d \cdot e \in \nu_0(\phi)] & \Longleftrightarrow (7.1(iii)) \\
\forall e\,[e \in \nu_0(\sigma) \Rightarrow \uparrow_{\mathrm{E}}\{\psi \mid \exists \rho \in e\,[\rho{\rightarrow}\psi \in d]\} \in \nu_0(\phi)] & \Longleftrightarrow \\
\forall e\,[\sigma \in e \Rightarrow \phi \in \uparrow_{\mathrm{E}}\{\psi \mid \exists \rho \in e\,[\rho{\rightarrow}\psi \in d]\}] & \Longleftrightarrow (\phi \in \mathcal{T}_{\mathrm{S}}) \\
\forall e\,[\sigma \in e \Rightarrow \exists \rho \in e\,[\rho{\rightarrow}\phi \in d]] & \Longleftrightarrow (\Rightarrow\text{: take } e = \uparrow_{\mathrm{E}}\sigma) \\
\sigma{\rightarrow}\phi \in d & \Longleftrightarrow \\
d \in \nu_0(\sigma{\rightarrow}\phi) &
\end{array}$$

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

We start by showing a soundness result.

Theorem 7.10 (Soundness) *$\Gamma \vdash_{\mathrm{E}} M:\sigma \Rightarrow \Gamma \vDash_{\mathrm{S}} M:\sigma$.*

Proof: By Definition 3.14, for all $\mathcal{M},\xi,v$, if $\mathcal{M},\xi,v \vDash_{\mathrm{S}} \Gamma$ then $\mathcal{M},\xi,v \vDash_{\mathrm{S}} M:\sigma$. This then means that, if $\mathcal{M},\xi,v \vDash x{:}\rho$ for every $x{:}\rho \in \Gamma$, then $\mathcal{M},\xi,v \vDash_{\mathrm{S}} M:\sigma$; so, we need to show: if $[\![x]\!]^{\mathcal{M}}_{\xi} \in v(\rho)$ for every $x{:}\rho \in \Gamma$, then $[\![M]\!]^{\mathcal{M}}_{\xi} \in v(\sigma)$.

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

(Ax): Then $\Gamma \vdash_{\mathrm{E}} x:\phi$, so there exists $x{:}\rho \in \Gamma$ such that $\rho \leq_{\mathrm{E}} \phi$. Assume $[\![x]\!]_{\xi} \in v(\rho)$, then, by Lemma 7.8, $[\![x]\!]_{\xi} \in v(\phi)$.

$({\rightarrow}I)$: Then $\sigma = \alpha{\rightarrow}\phi$, so $\Gamma \vdash_{\mathrm{E}} \lambda y.M':\alpha{\rightarrow}\phi$, and also $\Gamma,y{:}\alpha \vdash_{\mathrm{E}} M':\phi$. Let $e \in v(\alpha)$.

$$\begin{array}{ll}
\forall x{:}\tau \in \Gamma\cup\{y{:}\alpha\}\,[\,[\![x]\!]_{\xi(e/y)} \in v(\phi)] \;\&\; \Gamma,y{:}\alpha \vdash_{\mathrm{E}} M':\phi & \Rightarrow (IH) \\
[\![M']\!]_{\xi(e/y)} \in v(\phi) & \Rightarrow (7.6) \\
\{\psi \mid \Gamma_{\xi(e/y)} \vdash_{\mathrm{E}} M':\psi\} \in v(\phi) & \Rightarrow ({\rightarrow}I) \\
\uparrow_{\mathrm{E}}\{\psi \mid \exists \gamma \in e\,[\Gamma_{\xi} \vdash_{\mathrm{E}} \lambda y.M':\gamma{\rightarrow}\psi]\} \in v(\phi) & \Rightarrow \\
\uparrow_{\mathrm{E}}\{\psi \mid \exists \gamma \in e\,[\gamma{\rightarrow}\psi \in \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} \lambda y.M':\rho\}]\} \in v(\phi) & \Rightarrow (7.1(iii)) \\
\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} \lambda y.M':\rho\} \cdot e \in v(\phi). &
\end{array}$$

So, for all $e \in v(\alpha)$, we have shown that $\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} \lambda y.M':\rho\} \cdot e \in v(\phi)$, so, by Definition 3.13, we get $\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} \lambda y.M':\rho\} \in v(\alpha{\rightarrow}\phi)$.

$({\rightarrow}E)$: Then $\sigma = \phi$, $M \equiv PQ$, and there exists τ such that $\Gamma \vdash_{\mathrm{E}} P:\tau{\rightarrow}\phi$ and $\Gamma \vdash_{\mathrm{E}} Q:\tau$.

$$\begin{array}{ll}
\forall x{:}\tau \in \Gamma\,[\,[\![x]\!]_{\xi} \in v(\tau)] \;\&\; \Gamma \vdash_{\mathrm{E}} P:\tau{\rightarrow}\phi \;\&\; \Gamma \vdash_{\mathrm{E}} Q:\tau & \Rightarrow (IH) \\
[\![P]\!]_{\xi} \in v(\tau{\rightarrow}\phi) \;\&\; [\![Q]\!]_{\xi} \in v(\tau) & \Rightarrow (7.6) \\
\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} P:\rho\} \in v(\tau{\rightarrow}\phi) \;\&\; \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} Q:\rho\} \in v(\tau) & \Rightarrow ({\rightarrow}I) \\
\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} P:\rho\} \in \{d \mid \forall e \in v(\tau)\,[d \cdot e \in v(\phi)]\} \;\&\; \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} Q:\rho\} \in v(\tau) & \Rightarrow \\
\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} P:\rho\} \cdot \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} Q:\rho\} \in v(\phi) & \Rightarrow (7.1(iii)) \\
\uparrow_{\mathrm{E}}\{\psi \mid \exists \mu \in \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} Q:\rho\}\,[\mu{\rightarrow}\psi \in \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} P:\rho\}]\} \in v(\phi) & \Rightarrow \\
\uparrow_{\mathrm{E}}\{\psi \mid \exists \mu\,[\Gamma_{\xi} \vdash_{\mathrm{E}} Q:\nu \;\&\; \Gamma_{\xi} \vdash_{\mathrm{E}} P:\mu{\rightarrow}\psi]\} \in v(\phi) & \Rightarrow ({\rightarrow}E) \\
\uparrow_{\mathrm{E}}\{\psi \mid \Gamma_{\xi} \vdash_{\mathrm{E}} PQ:\psi\} \in v(\phi) & \Rightarrow \\
\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} PQ:\rho\} \in v(\phi). &
\end{array}$$

$(\cap I)$: Then $\sigma = \phi_1\cap\cdots\cap\phi_n$, and, for $i \in \underline{n}$, $\Gamma \vdash_{\mathrm{E}} M:\phi_i$.

$$\begin{array}{lll}
\text{Then:} & \forall x{:}\tau \in \Gamma\,[\,[\![x]\!]_{\xi} \in v(\tau)] \;\&\; \forall i \in \underline{n}\,[\Gamma \vdash_{\mathrm{E}} M:\phi_i] & \Rightarrow (IH) \\
& \forall i \in \underline{n}\,[\{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} M:\rho\} \in v(\phi_i)] & \Rightarrow \\
& \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} M:\rho\} \in v(\phi_1) \cap\cdots\cap v(\phi_n) & \Rightarrow (3.13) \\
& \{\rho \mid \Gamma_{\xi} \vdash_{\mathrm{E}} M:\rho\} \in v(\phi_1\cap\cdots\cap\phi_n). \;■ &
\end{array}$$

We need the following lemma in the proof below.

Lemma 7.11 i) $\Gamma \vdash_{\mathrm{E}} M:\sigma \Longleftrightarrow \Gamma_{\xi_{\Gamma}} \vdash_{\mathrm{E}} M:\sigma$.

ii) $\mathcal{F}_{\mathrm{E}},\xi_{\Gamma},\nu_0 \vDash_{\mathrm{S}} \Gamma$.

Proof: i) Because, for every x, $\xi_{\Gamma}(x)$ is a filter.

ii) $x{:}\sigma \in \Gamma \Rightarrow (i) \quad \sigma \in \{\tau \mid \Gamma_{\xi_\Gamma} \vdash_{\mathrm{E}} x{:}\tau\} \Rightarrow \sigma \in [\![x]\!]_{\xi_\Gamma}$.
So $[\![x]\!]_{\xi_\Gamma} \in \{d \in \mathcal{F}_{\mathrm{E}} \mid \sigma \in d\} = \nu_0(\sigma)$. ■

Since the interpretation of terms by their derivable types gives a λ-model, the following corollary is immediate and an alternative proof for Theorem 4.10.

Corollary 7.12 *If $M =_\beta N$ and $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$, then $\Gamma \vdash_{\mathrm{E}} N{:}\sigma$.*

Proof: Since $\mathcal{F}_{\mathrm{E}}$ is a λ-model, if $M =_\beta N$, then $[\![M]\!]_\xi = [\![N]\!]_\xi$, for any ξ, and, so $\{\sigma \mid \Gamma_\xi \vdash_{\mathrm{E}} M{:}\sigma\} = \{\sigma \mid \Gamma_\xi \vdash_{\mathrm{E}} N{:}\sigma\}$; then, by Lemma 7.11(*i*), $\{\sigma \mid \Gamma \vdash_{\mathrm{E}} M{:}\sigma\} = \{\sigma \mid \Gamma \vdash_{\mathrm{E}} N{:}\sigma\}$. ■

We can now show our completeness result.

Theorem 7.13 (Completeness) *$\Gamma \models_{\mathrm{S}} M{:}\sigma \Rightarrow \Gamma \vdash_{\mathrm{E}} M{:}\sigma$.*

Proof:
$$\begin{array}{ll}
\Gamma \models_{\mathrm{S}} M{:}\sigma & \Rightarrow (3.14, 7.11(ii)\ \&\ 7.9) \\
\mathcal{F}_{\mathrm{E}}, \xi_\Gamma, \nu_0 \models_{\mathrm{S}} M{:}\sigma & \Rightarrow (3.14) \\
[\![M]\!]_{\xi_\Gamma} \in \nu_0(\sigma) & \Rightarrow (7.7) \\
\sigma \in [\![M]\!]_{\xi_\Gamma} & \Rightarrow (7.6) \\
\Gamma_{\xi_B} \vdash_{\mathrm{E}} M{:}\sigma & \Rightarrow (7.11(i)) \\
\Gamma \vdash_{\mathrm{E}} M{:}\sigma. \ ■ &
\end{array}$$

7.3 Completeness for '$\vdash_{\mathrm{S}}$'

[3] also shows that strict type assignment is sound and complete with respect to inference type semantics: $\Gamma \models M{:}\sigma \iff \Gamma \vdash_{\mathrm{S}} M{:}\sigma$. In order to achieve that, we need to do a bit more work than in Section 7.2.

First of all, strict filters are defined as BCD filter or essential filters, but using '$\leq_{\mathrm{S}}$' rather than '$\leq$' or '$\leq_{\mathrm{E}}$'; the strict filter model $\mathcal{F}_{\mathrm{S}}$ is then defined as in Definition 7.1, but using '$\uparrow_{\mathrm{S}}$', the strict filter generator, to define application on strict filters and the domain constructor G. The construction then follows the lines of Section 7.1 and 7.2, but for the part where the type interpretation plays a role.

Notice that the filter λ-models $\mathcal{F}_{\mathrm{S}}$ and $\mathcal{F}_{\mathrm{E}}$ are not isomorphic as complete lattices, since, for example, in $\mathcal{F}_{\mathrm{E}}$ the filter $\uparrow(\sigma\cap\tau)\to\sigma$ is contained in $\uparrow\sigma\to\sigma$, but in $\mathcal{F}_{\mathrm{S}}$ the filter $\uparrow_{\mathrm{S}}(\sigma\cap\tau)\to\sigma$ is not contained in $\uparrow_{\mathrm{S}}\sigma\to\sigma$. Moreover, they are not isomorphic as λ-models since in $\mathcal{F}_{\mathrm{E}}$ the meaning of $(\lambda xy.xy)$ is contained in the meaning of $(\lambda x.x)$, while this does not hold in $\mathcal{F}_{\mathrm{S}}$.

Example 7.14 Notice that

$$\begin{aligned}
[\![\lambda xy.xy]\!]^{\mathcal{F}_{\mathrm{S}}} &= \uparrow_{\mathrm{S}}\{\rho\to\alpha\to\phi \mid \exists \sigma' \, [\rho \leq_{\mathrm{S}} \sigma'\to\phi \ \&\ \sigma \leq_{\mathrm{S}} \sigma']\} \\
[\![\lambda x.x]\!]^{\mathcal{F}_{\mathrm{S}}} &= \uparrow_{\mathrm{S}}\{\sigma\to\psi \mid \sigma \leq_{\mathrm{S}} \psi\}
\end{aligned}$$

and that $(\alpha\to\beta)\to\beta\cap\gamma\to\beta \in [\![\lambda xy.xy]\!]^{\mathcal{F}_{\mathrm{S}}}$, but $(\alpha\to\beta)\to\beta\cap\gamma\to\beta \notin [\![\lambda x.x]\!]^{\mathcal{F}_{\mathrm{S}}}$.

Another difference is that, while the analogue of G in $\mathcal{F}_{\mathrm{E}}$ chooses the minimal representative of functions, this is not the case in $\mathcal{F}_{\mathrm{S}}$. Moreover, it is straightforward to show that $\mathcal{F}_{\mathrm{S}}$ is equivalent to Engeler's model $\mathcal{D}_A$.

In [3], first it is shown that the map ν_0 (Definition 7.7) is an inference type interpretation.

Theorem 7.15 *The map $\nu_0(\sigma) = \{d \in \mathcal{F}_{\mathrm{S}} \mid \sigma \in d\}$ is an inference type interpretation.*

Proof: Observe that the neutral element ε in $\mathcal{F}_{\mathrm{S}}$ is $[\![\lambda xy.xy]\!]^{\mathcal{F}_{\mathrm{S}}}$. We check the conditions of 3.13.

i) $$\begin{array}{ll}
\forall e\,[e \in \nu_0(\sigma) \Rightarrow d \cdot e \in \nu_0(\phi)] & \Rightarrow \\
\forall e\,[e \in \nu_0(\sigma) \Rightarrow \varepsilon \cdot d \cdot e \in \nu_0(\phi)] & \Rightarrow (\textit{take } e = \uparrow_{\text{E}}\sigma) \\
\phi \in \varepsilon \cdot d \cdot \uparrow_{\text{S}}\sigma & \Rightarrow \\
\exists \rho \in \uparrow_{\text{S}}\sigma, \alpha \in d, \beta\, [\alpha \leq_{\text{S}} \beta \to \phi \;\&\; \rho \leq_{\text{S}} \beta] & \Rightarrow \\
\exists \alpha \in d, \beta\, [\alpha \leq_{\text{S}} \beta \to \phi \;\&\; \sigma \leq_{\text{S}} \beta] & \Rightarrow \\
\sigma \to \phi \in \uparrow_{\text{S}}\{\rho \to \psi \mid \exists \alpha \in d, \beta\, [\alpha \leq_{\text{S}} \beta \to \psi \;\&\; \rho \leq_{\text{S}} \beta]\} & \Rightarrow \\
\sigma \to \phi \in \uparrow_{\text{S}}\{\beta \mid \exists \alpha \in d\, [\alpha \to \beta \in \varepsilon]\} & \Rightarrow \\
\varepsilon \cdot d \in \nu_0(\sigma \to \phi). &
\end{array}$$

ii) Easy.

iii) Trivial. ■

Notice that although $\nu_0(\sigma\cap\tau) = \nu_0(\tau\cap\sigma)$, the sets $\nu_0((\sigma\cap\tau)\to\sigma)$ and $\nu_0((\tau\cap\sigma)\to\sigma)$ are incompatible. We can only show that both contain

$$\{\epsilon \cdot d \mid \forall e\,[e \in \nu_0(\sigma)\cap\nu_0(\tau) \Rightarrow d \cdot e \in \nu_0(\sigma)]\}$$

and are both contained in

$$\{d \mid \forall e\,[e \in \nu_0(\sigma)\cap\nu_0(\tau) \Rightarrow d \cdot e \in \nu_0(\sigma)]\}.$$

However, it is not difficult to prove that $\varepsilon \cdot \uparrow(\sigma\cap\tau)\to\sigma = \varepsilon \cdot \uparrow(\tau\cap\sigma)\to\sigma$, so the filters $\uparrow(\sigma\cap\tau)\to\sigma$ and $\uparrow(\tau\cap\sigma)\to\sigma$ represent the same function.

Using the fact that ν_0 is an inference type interpretation, in a way similar to that of the previous section, [3] shows $\Gamma \vdash_{\text{S}} M{:}\sigma \iff \Gamma \models M{:}\sigma$ (remark that the double turnstile is not subscripted).

8 Strong normalisation result for the system without ω

The other well-know result '*$\Gamma \vdash_{\text{E}} M{:}\sigma$ without using $\omega \iff M$ is strongly normalisable*' also holds for '$\vdash_{\text{E}}$', but needs a separate proof in that it is not a consequence of the Approximation Theorem 6.9. The proof for this property in [3] for '$\vdash_{\text{BCD}}$' follows very much the structure of the proof of Theorem 6.9; the proof we give here is new, but still uses a notion of computability; an alternative proof appeared in [11] which will be presented in Section 9.4. Alternatively, see [10] for a proof of this property set within '$\vdash_{\text{S}}$', where it is a direct consequence of the result that cut-elimination is strongly normalisable; it is this technique that will be extended to '$\vdash_{\text{E}}$' in Section 9.

8.1 Intersection Type Assignment without ω

We will prove that the set of all terms typeable by the system without ω is the set of all strongly normalisable terms. We start by defining that notion of type assignment formally.

Definition 8.1 *i*) The set of *strict ω-free intersection types*, ranged over by σ, τ, … and its subset of *strict (intersection) ω-free types* ranged over by $\phi, \psi, \ldots$, are defined through the grammar:

$$\begin{array}{rcl}
\phi, \psi & ::= & \varphi \mid \sigma \to \psi \\
\sigma, \tau & ::= & \phi_1 \cap \cdots \cap \phi_n \quad (n \geq 1)
\end{array}$$

We will use $\mathcal{T}_{\not\omega}$ for the set of all ω-free types. (Notice that the only difference between this definition and Definition 4.1 is that $n \geq 1$ in $\phi_1 \cap \cdots \cap \phi_n$ rather than $n \geq 0$.)

ii) On $\mathcal{T}_{\not\omega}$ the relation '$\leq_E$' is as defined in Definition 4.1, except for the second alternative.

$$\begin{array}{rcll} \forall i \in \underline{n}\ [\,\phi_1\cap\cdots\cap\phi_n & \leq_E & \phi_i\,] & (n\geq 1) \\ \forall i \in \underline{n}\ [\sigma \leq_E \phi_i] & \Rightarrow & \sigma \leq_E \phi_1\cap\cdots\cap\phi_n & (n\geq 1) \\ \rho\leq_E\sigma \ \&\ \phi\leq_E\psi & \Rightarrow & \sigma\rightarrow\phi\leq_E\rho\rightarrow\psi & \end{array}$$

and the relation '$\sim_E$' is the equivalence relation generated by '$\leq_E$'; the relations '$\leq_E$' and '$\sim_E$' are extended to contexts as before.

iii) We write $\Gamma \vdash_{\not\omega} M{:}\sigma$ if this statement is derivable from Γ, using only ω-free types and the derivation rules of '$\vdash_E$' (so, for rule $(\cap I)$, $n \geq 1$).

For the ω-free system, the following properties hold:

Lemma 8.2
$$\begin{array}{lcl} \Gamma \vdash_{\not\omega} x{:}\sigma & \Longleftrightarrow & \exists \rho \in \mathcal{T}\ [x{:}\rho\in\Gamma \ \&\ \rho\leq_E\sigma] \\ \Gamma \vdash_{\not\omega} MN{:}\phi & \Longleftrightarrow & \exists\tau\ [\Gamma \vdash_{\not\omega} M{:}\tau\rightarrow\phi \ \&\ \Gamma \vdash_{\not\omega} N{:}\tau] \\ \Gamma \vdash_{\not\omega} \lambda x.M{:}\phi & \Longleftrightarrow & \exists\rho,\psi\ [\phi=\rho\rightarrow\psi \ \&\ \Gamma,x{:}\rho \vdash_{\not\omega} M{:}\psi] \\ \Gamma \vdash_{\not\omega} M{:}\sigma \ \&\ \Gamma'\leq_E\Gamma \ \&\ \Gamma'\ \omega\text{-free} & \Rightarrow & \Gamma' \vdash_{\not\omega} M{:}\sigma \\ \mathcal{D} :: \Gamma \vdash_{\not\omega} M{:}\sigma & \Rightarrow & \mathcal{D} :: \Gamma \vdash_E M{:}\sigma \end{array}$$

Proof: Easy. ■

The following lemma is needed in the proof of Theorem 8.7.

Lemma 8.3 *If A is $\perp$-free, then there are Γ, and ϕ, such that $\Gamma \vdash_{\not\omega} A{:}\phi$.*

Proof: By Lemma 6.16, and the observation made after the proof of that lemma. ■

8.2 Strong Normalisation implies Typeability

The following lemma shows a subject expansion result for the ω-free system.

Lemma 8.4 *If $\Gamma \vdash_{\not\omega} M[N/x]{:}\sigma$ and $\Gamma \vdash_{\not\omega} N{:}\rho$, then $\Gamma \vdash_{\not\omega} (\lambda x.M)N{:}\sigma$.*

Proof: As usual, we focus on the case that $\sigma=\phi\in\mathcal{T}_s$, the case that σ is an intersection is just a generalisation. We can assume that x does not occur in Γ, and proceed by induction on the structure of M.

$(M\equiv x)$: $\Gamma \vdash_{\not\omega} N{:}\phi \Rightarrow \Gamma \vdash_{\not\omega} (\lambda x.x)N{:}\phi$

$(M\equiv y\neq x)$:
$$\begin{array}{ll} \Gamma \vdash_{\not\omega} y{:}\phi \ \&\ \Gamma \vdash_{\not\omega} N{:}\rho & \Rightarrow (8.2\ \&\ (\rightarrow I)) \\ \Gamma \vdash_{\not\omega} \lambda x.y{:}\rho\rightarrow\phi \ \&\ \Gamma \vdash_{\not\omega} N{:}\rho & \Rightarrow (\rightarrow E) \\ \Gamma \vdash_{\not\omega} (\lambda x.y)N{:}\phi. & \end{array}$$

$(M\equiv\lambda y.M')$: Then $(\lambda y.M')[N/x]\equiv\lambda y.(M'[N/x])$, and $\phi=\delta\rightarrow\psi$.
$$\begin{array}{ll} \Gamma \vdash_{\not\omega} \lambda y.(M'[N/x]){:}\delta\rightarrow\psi \ \&\ \Gamma \vdash_{\not\omega} N{:}\rho & \Rightarrow (\rightarrow I) \\ \Gamma,y{:}\delta \vdash_{\not\omega} M'[N/x]{:}\psi \ \&\ \Gamma \vdash_{\not\omega} N{:}\rho & \Rightarrow (IH) \\ \Gamma,y{:}\delta \vdash_{\not\omega} (\lambda x.M')N{:}\psi & \Rightarrow (\rightarrow E) \\ \exists\tau\ [\Gamma,y{:}\delta \vdash_{\not\omega} \lambda x.M'{:}\tau\rightarrow\psi \ \&\ \Gamma,y{:}\delta \vdash_{\not\omega} N{:}\tau] & \Rightarrow ((\rightarrow I)\ \&\ y\notin fv(N)) \\ \exists\tau\ [\Gamma,y{:}\delta,x{:}\tau \vdash_{\not\omega} M'{:}\psi \ \&\ \Gamma \vdash_{\not\omega} N{:}\tau] & \Rightarrow (\rightarrow I) \\ \exists\tau\ [\Gamma \vdash_{\not\omega} \lambda xy.M'{:}\tau\rightarrow\delta\rightarrow\psi \ \&\ \Gamma \vdash_{\not\omega} N{:}\tau] & \Rightarrow (\rightarrow E) \\ \Gamma \vdash_{\not\omega} (\lambda xy.M')N{:}\delta\rightarrow\psi & \end{array}$$

$(M\equiv M_1M_2)$: Then $(M_1M_2)[N/x]\equiv M_1[N/x]M_2[N/x]$.

$$\begin{array}{ll}
\Gamma \vdash_{\not\omega} M_1[N/x]M_2[N/x] : \phi \ \&\ \Gamma \vdash_{\not\omega} N : \rho & \Rightarrow (\to E) \\
\exists \tau\ [\Gamma \vdash_{\not\omega} M_1[N/x] : \tau\to\phi \ \&\ \Gamma \vdash_{\not\omega} M_2[N/x] : \tau] \ \&\ \Gamma \vdash_{\not\omega} N : \rho & \Rightarrow (IH) \\
\exists \tau\ [\Gamma \vdash_{\not\omega} (\lambda x.M_1)N : \tau\to\sigma \ \&\ \Gamma \vdash_{\not\omega} (\lambda x.M_2)N : \tau] & \Rightarrow ((\to E)\ \&\ (\to I)) \\
\multicolumn{2}{l}{\exists \rho_1,\rho_2,\tau\ [\Gamma, x{:}\rho_1 \vdash_{\not\omega} M_1 : \tau\to\phi \ \&\ \Gamma \vdash_{\not\omega} N : \rho_1 \ \&\ \Gamma, x{:}\rho_2 \vdash_{\not\omega} M_2 : \tau \ \&\ \Gamma \vdash_{\not\omega} N : \rho_2]} \\
 & \Rightarrow ((\cap I)\ \&\ 8.2) \\
\exists \rho_1,\rho_2\ [\Gamma, x{:}\rho_1\cap\rho_2 \vdash_{\not\omega} M_1M_2 : \phi \ \&\ \Gamma \vdash_{\not\omega} N : \rho_1\cap\rho_2] & \Rightarrow (\to I) \\
\exists \rho_1,\rho_2\ [\Gamma \vdash_{\not\omega} \lambda x.(M_1M_2) : \rho_1\cap\rho_2\to\phi \ \&\ \Gamma \vdash_{\not\omega} N : \rho_1\cap\rho_2] & \Rightarrow (\to E) \\
\Gamma \vdash_{\not\omega} (\lambda x.(M_1M_2))N : \phi]. \ \blacksquare &
\end{array}$$

Notice that the condition $\Gamma \vdash_{\not\omega} N : \rho$ in the formulation of the lemma is essential and is used in part $M \equiv y \neq x$. As a counter example, take the two λ-terms $\lambda yz.(\lambda b.z)(yz)$ and $\lambda yz.z$. Notice that the first strongly reduces to the latter. We know that

$$z{:}\sigma, y{:}\tau \vdash_{\not\omega} z : \sigma$$

but it is impossible to give a derivation for $(\lambda b.z)(yz) : \sigma$ from the same context without using ω. This is caused by the fact that we can only type $(\lambda b.z)(yz)$ in the system without ω from a context in which the predicate for y is an arrow type. We can, for example, derive

$$\dfrac{\dfrac{\overline{y{:}\sigma\to\tau, z{:}\sigma, b{:}\tau \vdash z : \sigma}}{y{:}\sigma\to\tau, z{:}\sigma \vdash \lambda b.z : \tau\to\sigma}\,(\to I) \quad \dfrac{\overline{y{:}\sigma\to\tau, z{:}\sigma \vdash y : \sigma\to\tau} \quad \overline{y{:}\sigma\to\tau, z{:}\sigma \vdash z : \sigma}}{y{:}\sigma\to\tau, z{:}\sigma \vdash yz : \tau}\,(\to E)}{y{:}\sigma\to\tau, z{:}\sigma \vdash (\lambda b.z)(yz) : \sigma}\,(\to E)$$

We can therefore only state that we can derive $\vdash_{\not\omega} \lambda yz.(\lambda b.z)(yz) : (\sigma\to\tau)\to\sigma\to\sigma$ and $\vdash_{\not\omega} \lambda yz.z : \tau\to\sigma\to\sigma$, but we are not able to give a derivation without ω for the statement $\vdash_{\not\omega} \lambda yz.(\lambda b.z)(yz) : \tau\to\sigma\to\sigma$. So the type assignment without ω is not closed for β-equality. Notice that in '$\vdash_{\text{E}}$' we can derive:

$$\dfrac{\dfrac{\dfrac{\dfrac{\overline{y{:}\tau, z{:}\sigma, b{:}\omega \vdash z : \sigma}}{y{:}\tau, z{:}\sigma \vdash \lambda b.z : \omega\to\sigma}\,(\to I) \quad \dfrac{}{y{:}\tau, z{:}\sigma \vdash yz : \omega}\,(\cap I)}{y{:}\tau, z{:}\sigma \vdash (\lambda b.z)(yz) : \sigma}\,(\to E)}{y{:}\tau \vdash \lambda z.(\lambda b.z)(yz) : \sigma\to\sigma}\,(\to I)}{\vdash \lambda yz.(\lambda b.z)(yz) : \tau\to\sigma\to\sigma}\,(\to I)$$

We will now show that all strongly normalisable terms are typeable in '$\vdash_{\not\omega}$'. The proof of the crucial lemma for this result as presented below (Lemma 8.6) is due to Betti Venneri, adapted to '$\leq_{\text{E}}$', and goes by induction on the left-most outer-most reduction path.

Definition 8.5 An occurrence of a redex $\text{R} = (\lambda x.P)Q$ in a term M is called the *left-most outer-most redex of* M ($lor(M)$), if and only if:

i) there is no redex R' in M such that $\text{R}' = \text{C}[\text{R}]$ (*outer-most*);

ii) there is no redex R' in M such that $M = \text{C}_0[\text{C}_1[\text{R}']\text{C}_2[\text{R}]]$ (*left-most*).

$M \to_{lor} N$ is used to indicate that M reduces to N by contracting $lor(M)$.

The following lemma formulates a subject expansion result for '$\vdash_{\not\omega}$' with respect to left-most outer-most reduction. A proof for this property in the context of '$\vdash_{\text{S}}$' appeared in [10].

Lemma 8.6 Let $M \to_{lor} N$, $lor(M) = (\lambda x.P)Q$, $\Gamma \vdash_{\not\omega} N : \sigma$, *and* $\Gamma' \vdash_{\not\omega} Q : \tau$, *then there exists* Γ_1, ρ *such that* $\sigma \leq_{\text{E}} \rho$, *and* $\Gamma_1 \vdash_{\not\omega} M : \rho$.

Proof: By induction on the structure of types, of which only the part $\sigma \in \mathcal{T}_{\mathrm{S}}$ will be shown. This part follows by induction on the structure of terms.

Note that $M \equiv \lambda x_1 \cdots x_k.VP_1 \cdots P_n$ $(k,n \geq 0)$, where either

i) V is a redex $(\lambda y.P)Q$, so $lor(M) = V$ and $N \equiv \lambda x_1 \cdots x_k.(P[Q/y])P_1 \cdots P_n$, or
ii) $V \equiv y$, so there is an $j \in \underline{n}$ such that $lor(M) = lor(P_j)$, and $P_j \rightarrow_{lor} P'$, and $N \equiv \lambda x_1 \cdots x_k.yP_1 \cdots P' \cdots P_n$

In either case, we have, by Lemma 8.2, that there are α_i $(i \in \underline{k})$, γ_j $(j \in \underline{n})$, and ϕ such that

$$\sigma = \alpha_1 \rightarrow \cdots \rightarrow \alpha_k \rightarrow \phi, \Gamma_0 \vdash_{\not\omega} V' : \gamma_1 \rightarrow \cdots \rightarrow \gamma_n \rightarrow \phi, \text{ and } \Gamma_0 \vdash_{\not\omega} P_i : \gamma_i \ (i \in \underline{n}),$$

where $\Gamma_0 = \Gamma, x_1{:}\alpha_1, \ldots, x_k{:}\alpha_k$, and V' is either $P[Q/y]$ or y. So:

i) $V' \equiv P[Q/y]$. Let $\Gamma_1 = \Gamma'$, then, by Lemma 8.4,

$$\cap\{\Gamma_0, \Gamma_1\} \vdash_{\not\omega} (\lambda y.P)Q : \gamma_1 \rightarrow \cdots \rightarrow \gamma_n \rightarrow \phi.$$

ii) $V' \equiv y$. Then, by induction, there are Γ', ρ such that $\gamma_j \leq_{\mathrm{E}} \rho$, and $\Gamma' \vdash_{\not\omega} P_j : \rho$. Take $\Gamma_1 = \Gamma', y{:}\gamma_1 \rightarrow \cdots \rho \cdots \rightarrow \gamma_n \rightarrow \phi$, then

$$\cap\{\Gamma_0, \Gamma_1\} \vdash_{\not\omega} y : \gamma_1 \rightarrow \cdots \rho \cdots \rightarrow \gamma_n \rightarrow \phi.$$

In either case, $\cap\{\Gamma_0, \Gamma_1\} \vdash_{\not\omega} VP_1 \cdots P_n : \phi$. Let, for $i \in \underline{k}$, $x_i{:}\beta_i \in \cap\{\Gamma_0, \Gamma_1\}$ then

$$\cap\{\Gamma_0, \Gamma_1\} \backslash x_1, \ldots, x_k \vdash_{\not\omega} \lambda x_1 \cdots x_k.yP_1 \cdots P_n : \beta_1 \rightarrow \cdots \rightarrow \beta_k \rightarrow \phi,$$

and, in particular, since $\beta_i \leq_{\mathrm{E}} \alpha_i$ for $i \in \underline{n}$, $\alpha_1 \rightarrow \cdots \rightarrow \alpha_k \rightarrow \phi \leq_{\mathrm{E}} \beta_1 \rightarrow \cdots \rightarrow \beta_k \rightarrow \phi$. ■

We can now show that all strongly normalisable terms are typeable in '$\vdash_{\not\omega}$'.

Theorem 8.7 *If M is strongly normalisable, then $\Gamma \vdash_{\not\omega} M : \sigma$ for some Γ, σ.*

Proof: By induction on the maximum of the lengths of reduction sequences for a strongly normalisable term to its normal form (denoted by $\#(M)$).

i) If $\#(M) = 0$, then M is in normal form, and by Lemma 8.3, there exist Γ and ϕ such that $\Gamma \vdash_{\not\omega} M : \phi$.
ii) If $\#(M) \geq 1$, so M contains a redex, then let $M \rightarrow_{lor} N$ by contracting $(\lambda x.P)Q$. Then $\#(N) < \#(M)$, and $\#(Q) < \#(M)$ (since Q is a proper subterm of a redex in M), so by induction $\Gamma \vdash_{\not\omega} N : \sigma$ and $\Gamma' \vdash_{\not\omega} Q : \tau$, for some Γ, Γ', σ, and τ. Then, by Lemma 8.6, there exist Γ_1, ρ such that $\Gamma_1 \vdash_{\not\omega} M : \rho$. ■

8.3 Strong normalisation

We shall prove that, when ω is removed from the system, every typeable term is strongly normalisable. This will be done using Tait's method. We use $SN(M)$ to express that M is strongly normalisable, and $\mathcal{SN} = \{M \mid SN(M)\}$.

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

Fact 8.8 *i)* If $SN(x\overrightarrow{M_i})$ and $SN(N)$, then $SN(x\overrightarrow{M_i}N)$.
ii) If $SN(M[N/x]\overrightarrow{P})$ and $SN(N)$, then $SN((\lambda x.M)N\overrightarrow{P})$.

Definition 8.9 We define the set $Red(\rho)$ inductively over types by:

$$\begin{array}{rcl} Red(\varphi) & = & \mathcal{SN} \\ Red(\sigma \rightarrow \phi) & = & \{M \mid \forall N\, [N \in Red(\sigma) \Rightarrow MN \in Red(\phi)]\} \\ Red(\phi_1 \cap \cdots \cap \phi_n) & = & \bigcap_{1 \leq i \leq n} Red(\phi_i). \end{array}$$

Notice that this notion of computability is not defined in terms of typeability at all; this is the main difference between the structure of the proof here and that presented in [3].

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

Lemma 8.10 *For all* ρ,

i) $Red\,(\rho) \subseteq \mathcal{SN}$.

ii) $SN\,(x\overrightarrow{N}) \Rightarrow x\overrightarrow{N} \in Red\,(\rho)$.

Proof: By simultaneous induction on the structure of types, using Definition 8.9.

i) (φ): Immediate.

$$\begin{array}{lll} (\sigma{\rightarrow}\phi): & M \in Red\,(\sigma{\rightarrow}\phi) & \Rightarrow (IH(ii)) \\ & x \in Red\,(\sigma) \;\&\; M \in Red\,(\sigma{\rightarrow}\phi) & \Rightarrow (8.9) \\ & Mx \in Red\,(\phi) & \Rightarrow (IH(i)) \\ & Mx \in \mathcal{SN} & \Rightarrow \\ & M \in \mathcal{SN}. & \end{array}$$

$(\phi_1{\cap}\cdots{\cap}\phi_n)$: $M \in Red\,(\phi_1{\cap}\cdots{\cap}\phi_n) \Rightarrow (8.9)\; M \in Red\,(\phi_i) \Rightarrow (IH(i))\; M \in \mathcal{SN}$.

ii) (φ): $SN\,(x\overrightarrow{N}) \Rightarrow (8.9)\; x\overrightarrow{N} \in Red\,(\varphi)$.

$$\begin{array}{lll} (\sigma{\rightarrow}\phi): & SN\,(x\overrightarrow{N}) & \Rightarrow (8.9 \;\&\; IH(i)) \\ & P \in Red\,(\sigma) \Rightarrow SN\,(x\overrightarrow{N}) \;\&\; SN\,(P) & \Rightarrow (8.8(i)) \\ & P \in Red\,(\sigma) \Rightarrow SN\,(x\overrightarrow{N}P) & \Rightarrow (IH(ii)) \\ & P \in Red\,(\sigma) \Rightarrow x\overrightarrow{N}P \in Red\,(\phi) & \Rightarrow (8.9) \\ & x\overrightarrow{N} \in Red\,(\sigma{\rightarrow}\phi) & \end{array}$$

$$\begin{array}{lll} (\phi_1{\cap}\cdots{\cap}\phi_n): & SN\,(x\overrightarrow{N}) & \Rightarrow (IH(ii)) \\ & \forall\, i \in \underline{n}\; [\, x\overrightarrow{N} \in Red\,(\phi_i)\,] & \Rightarrow \\ & x\overrightarrow{N} \in \cap_n Red\,(\phi_i) & \Rightarrow (8.9) \\ & x\overrightarrow{N} \in Red\,(\phi_1{\cap}\cdots{\cap}\phi_n). \;\blacksquare & \end{array}$$

The following result, stating that all term-variables are reducible of any type, follows immediately from part (*ii*):

Corollary 8.11 *For all* x, ρ: $x \in Red\,(\rho)$.

We will now show that the reducibility predicate is closed for '$\leq_{\rm E}$'.

Lemma 8.12 *Take* σ *and* τ *such that* $\sigma \leq_{\rm E} \tau$. *Then* $Red\,(\sigma) \subseteq Red\,(\tau)$.

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

$(\phi_1{\cap}\cdots{\cap}\phi_n \leq_{\rm E} \phi_i\; (i \in \underline{n}))$: $Red\,(\phi_1{\cap}\cdots{\cap}\phi_n) = (8.9)\; \bigcap_{i \in n} Red\,(\phi_i) \subseteq Red\,(\phi_i)$.

$(\tau \leq_{\rm E} \phi_i\; (\forall i \in \underline{n}) \Rightarrow \tau \leq_{\rm E} \phi_1{\cap}\cdots{\cap}\phi_n)$: $M \in Red\,(\tau) \Rightarrow (IH)$

$M \in Red\,(\phi_i)\; (\forall\, i \in \underline{n}) \Rightarrow M \in \bigcap_{i \in n} Red\,(\phi_i) \Rightarrow (8.9)\; M \in Red\,(\phi_1{\cap}\cdots{\cap}\phi_n)$

$$\begin{array}{ll}
(\rho \leq_{\mathrm{E}} \sigma \ \& \ \phi \leq_{\mathrm{E}} \psi \Rightarrow \sigma \rightarrow \phi \leq_{\mathrm{E}} \rho \rightarrow \psi): \quad M \in Red \sigma \rightarrow \phi & \Rightarrow (8.9) \\
\quad (N \in Red(\sigma) \Rightarrow MN \in Red(\phi)) & \Rightarrow \\
\quad (N \in Red(\rho) \Rightarrow (IH) \ N \in Red(\sigma) \Rightarrow & \\
\qquad MN \in Red(\phi) \Rightarrow (IH) \ MN \in Red(\psi)) & \Rightarrow \\
\quad (N \in Red(\rho) \Rightarrow MN \in Red(\psi)) & \Rightarrow (8.9) \\
\quad M \in Red(\rho \rightarrow \psi). \ \blacksquare &
\end{array}$$

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

Lemma 8.13 $M[N/x]\vec{P} \in Red(\sigma) \ \& \ N \in Red(\rho) \Rightarrow (\lambda x.M)N\vec{P} \in Red(\sigma)$.

Proof: By induction on the structure of types.

$$\begin{array}{lll}
(\varphi): & M[N/x]\vec{P} \in Red(\varphi) \ \& \ N \in Red(\rho) & \Rightarrow (8.9) \\
& SN(M[N/x]\vec{P}) \ \& \ SN(N) & \Rightarrow (8.8(ii) \ \& \ 8.10(i)) \\
& SN((\lambda x.M)N\vec{P}) & \Rightarrow (8.9) \\
& (\lambda x.M)N\vec{P} \in Red(\varphi) &
\end{array}$$

$$\begin{array}{lll}
(\sigma \rightarrow \phi): & M[N/x]\vec{P} \in Red(\sigma \rightarrow \phi) \ \& \ N \in Red(\rho) & \Rightarrow (8.9) \\
& Q \in Red(\sigma) \Rightarrow M[N/x]\vec{P}Q \in Red(\phi) \ \& \ N \in Red(\rho) & \Rightarrow (IH) \\
& Q \in Red(\sigma) \Rightarrow (\lambda x.M)N\vec{P}Q \in Red(\phi) & \Rightarrow (8.9) \\
& (\lambda x.M)N\vec{P} \in Red(\sigma \rightarrow \phi) &
\end{array}$$

$(\phi_1 \cap \cdots \cap \phi_n)$: Directly by induction and Definition 8.9. ■

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

Theorem 8.14 (Replacement property) *Let $\Gamma = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$. If, for all $i \in \underline{n}$, $N_i \in Red(\mu_i)$, and $\Gamma \vdash_{\not\omega} M{:}\sigma$, then $M[\overrightarrow{N_i/x_i}] \in Red(\sigma)$.*

Proof: By induction on the structure of derivations.

(Ax): Then $M \equiv x_j$, for some $j \in \underline{n}$, $\mu_j \leq_{\mathrm{E}} \sigma$, and $M[\overrightarrow{N_i/x_i}] \equiv x_j[\overrightarrow{N_i/x_i}] \equiv N_j$. From $N_j \in Red(\mu_j)$, by Lemma 8.12, also $N_j \in Red(\sigma)$.

$(\rightarrow I)$: Then $M \equiv \lambda y.M'$, $\sigma = \rho \rightarrow \phi$, and $\Gamma, y{:}\rho \vdash_{\not\omega} M'{:}\phi$.

$$\begin{array}{ll}
\forall i \in \underline{n} \ [N_i \in Red(\mu_i)] \ \& \ \Gamma, y{:}\rho \vdash_{\not\omega} M'{:}\phi & \Rightarrow (IH) \\
N \in Red(\rho) \Rightarrow M'[\overrightarrow{N_i/x_i}, N/y] \in Red(\phi) & \Rightarrow (8.13) \\
N \in Red(\rho) \Rightarrow (\lambda y.M'[\overrightarrow{N_i/x_i}])N \in Red(\phi) & \Rightarrow (8.9) \\
(\lambda y.M')[\overrightarrow{N_i/x_i}] \in Red(\rho \rightarrow \phi). &
\end{array}$$

$(\rightarrow E)$, $(\cap I)$: Straightforward by induction and Definition 8.9. ■

We can now prove the main result.

Theorem 8.15 (Strong Normalisation) *Any term typeable in '$\vdash_{\not\omega}$' is strongly normalisable.*

Proof: Let $\Gamma = \{x_1{:}\mu_1, \ldots, x_n{:}\mu_n\}$ such that $\Gamma \vdash_{\not\omega} M{:}\sigma$. By Corollary 8.11, for all $i \in \underline{n}$, $x_i \in Red(\mu_i)$. Then, by 8.14, $M[\overrightarrow{x_i/x_i}] \in Red(\sigma)$, so M itself is reducible of type σ. Strong normalisation for M then follows from Lemma 8.10(i). ■

This property can be shown also for the ω-free version of '$\vdash_{\mathrm{S}}$', in a way similar to the proof above, but using '$\leq_{\mathrm{S}}$' rather than '$\leq_{\mathrm{E}}$'.

9 Strong normalisation for Derivation Reduction

In this section, we will define a notion of reduction on derivations of the essential type assignment system, and show this notion to be strongly normalisable, as well as that all other characterisation results are a consequence of this. The technique used is based on the notion of cut-elimination developed in collaboration with Fernàndez for Term Rewriting [12], which was later used for Combinator Systems in [13], and was also used for '$\vdash_{\rm S}$' in [10].

Strong normalisation of cut-elimination is a well established property in the area of logic and has been studied profoundly in the past. In the area of type assignment for the λ-calculus, the corresponding property is that of strong normalisation of derivation reduction (also called cut-elimination in, for example, [15]), which mimics the normal reduction on terms to which the types are assigned, and also this area has been well studied.

For intersection type assignment systems, proofs of strong normalisation of derivation reduction have at best been indirect, i.e. obtained through a mapping from the derivations into a logic by [50], where the property has been established before. Since in those logics the type-constant ω cannot be adequately mapped, the intersection systems studied in that way are ω-free. (There exists a logic that deals adequately with intersection and ω [32], but strong normalisation of cut-elimination has not yet been shown for it.) This section will present a proof for the property directly in the system itself. We will then show that all characterisation results are direct consequences.

The added complexity of intersection types implies that, unlike for ordinary systems of type assignment, there is a significant difference between derivation reduction and ordinary reduction (see the beginning of Section 9.2); not only because of the presence of the type-constant ω, unlike normal typed- or type assignment system, not every term-redex occurs with types in a derivation.

For the general idea, we will be contracting derivations structured like

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma,x{:}\rho \vdash P:\phi\end{array}}{\Gamma,x{:}\rho \vdash \lambda x.P:\rho\to\phi}\,(\to I) \qquad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash Q:\rho\end{array}}{\Gamma \vdash (\lambda x.P)Q:\phi}\,(\to E)$$

which will contract to

$$\begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash Q:\rho\\ \boxed{\quad\mathcal{D}_1\quad}\\ \Gamma \vdash P[Q/x]:\phi\end{array}$$

However, as we will illustrate below, this picture is incomplete.

As for the proof of this property in '$\vdash_{\rm S}$' in [10], it is very similar to what follows, with the exception of the dealings with the contra-variant '$\leq_{\rm E}$' relation (Definition 9.2, and Lemma 9.3 and 9.12).

9.1 Partial order on derivations

We will use the following short-hand notation for derivations.

Definition 9.1 *i)* $\mathcal{D} = \langle Ax \rangle :: \Gamma \vdash_{\rm E} x{:}\phi$ if $\mathcal{D}$ consists of nothing but an application of rule (Ax).

ii) $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_n,\cap I \rangle$, if and only if there are $\sigma_1,\ldots,\sigma_n$ such that $\mathcal{D}_i :: \Gamma \vdash_{\rm E} M{:}\phi_i$ for $i \in \underline{n}$, and $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\phi_1\cap\cdots\cap\phi_n$ is obtained from $\mathcal{D}_1,\ldots,\mathcal{D}_n$ by applying rule $(\cap I)$.

iii) $\mathcal{D} = \langle \mathcal{D}', \to I \rangle$, if and only if there are M', σ, ϕ such that $\mathcal{D}' :: \Gamma, x{:}\sigma \vdash_{\rm E} M' : \phi$, and $\mathcal{D} :: \Gamma \vdash_{\rm E} \lambda x.M' : \sigma \to \phi$ is obtained from $\mathcal{D}'$ by applying rule $(\to I)$.

iv) $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \to E \rangle$, if and only if there are P, Q, and σ, ϕ such that $\mathcal{D}_2 :: \Gamma \vdash_{\rm E} Q : \sigma$ and $\mathcal{D}_1 :: \Gamma \vdash_{\rm E} P : \sigma \to \phi$, and $\mathcal{D} :: \Gamma \vdash_{\rm E} PQ : \phi$ is obtained from $\mathcal{D}_1$ and $\mathcal{D}_2$ by applying rule $(\to E)$.

We will identify derivations that have the same structure in that they have the same rules applied in the same order (so are, apart from sub-terms typed with ω, derivations involving the same term); the types derived need not be the same.

We now extend the relation '$\leq_{\rm E}$' on types to derivations in '$\vdash_{\rm E}$'; this notion is pivotal for the proof of strong normalisation of derivation reduction, when we need to show that computability is closed for '$\leq_{\rm E}$'.

Definition 9.2 *i)* $\langle Ax \rangle :: \Gamma \vdash_{\rm E} x : \sigma \preceq \langle Ax \rangle :: \Gamma' \vdash_{\rm E} x : \sigma'$ for all Γ', s' with $\Gamma' \leq_{\rm E} \Gamma$, and $\sigma \leq_{\rm E} \sigma'$.

ii) $\langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I \rangle :: \Gamma \vdash_{\rm E} M : \phi_1 \cap \cdots \cap \phi_n \preceq \langle \mathcal{D}'_1, \ldots, \mathcal{D}'_m, \cap I \rangle :: \Gamma' \vdash_{\rm E} M : \cap_m \phi'_j$, if and only if for every $j \in \underline{m}$ there exists an $i \in \underline{n}$ such that $\mathcal{D}_i \preceq \mathcal{D}'_j$.

iii) $\langle \mathcal{D} :: \Gamma, x{:}\sigma \vdash_{\rm E} M : \phi, \to I \rangle :: \Gamma \vdash_{\rm E} \lambda x.M : \sigma \to \phi \preceq$
$\langle \mathcal{D}' :: \Gamma', x{:}\sigma' \vdash_{\rm E} M : \phi', \to I \rangle :: \Gamma' \vdash_{\rm E} \lambda x.M' : \sigma' \to \phi'$
if and only if $\mathcal{D} \preceq \mathcal{D}'$.

iv) Let $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma \vdash_{\rm E} P : \sigma \to \phi, \mathcal{D}_2 :: \Gamma \vdash_{\rm E} Q : \sigma, \to E \rangle :: \Gamma \vdash_{\rm E} PQ : \phi$. Then, for $\sigma' \leq_{\rm E} \sigma, \phi' \geq_{\rm E} \phi$, $\mathcal{D}'_1 \preceq \mathcal{D}_1$, $\mathcal{D}'_2 \succeq \mathcal{D}_2$ such that $\mathcal{D}'_1 :: \Gamma' \vdash_{\rm E} P : \sigma' \to \phi'$ and $\mathcal{D}'_2 :: \Gamma' \vdash_{\rm E} Q : \sigma'$:

$$\mathcal{D} \preceq \langle \mathcal{D}'_1 :: \Gamma' \vdash_{\rm E} P : \sigma' \to \phi', \mathcal{D}'_2 :: \Gamma' \vdash_{\rm E} Q : \sigma', \to E \rangle :: \Gamma' \vdash_{\rm E} PQ : \phi'.$$

Notice that '$\preceq$' is contra-variant in $(\to E)$.

The following is easy to show, generalises Lemma 4.5, and establishes the relation between '$\leq_{\rm E}$' on types and '$\preceq$' on derivations:

Lemma 9.3 i) If $\mathcal{D} :: \Gamma \vdash_{\rm E} M : \sigma$ and $\Gamma' \leq_{\rm E} \Gamma$, $\sigma \leq_{\rm E} \sigma'$, then there exists $\mathcal{D}' \succeq \mathcal{D}$ such that $\mathcal{D}' :: \Gamma' \vdash_{\rm E} M' : \sigma'$.

ii) If $\mathcal{D} :: \Gamma \vdash_{\rm E} M : \sigma \preceq \mathcal{D}' :: \Gamma' \vdash_{\rm E} M' : \sigma'$, then $\Gamma' \leq_{\rm E} \Gamma$, and $\sigma \leq_{\rm E} \sigma'$.

Proof: i) We separate two cases: We separate two cases: We separate two cases: We separate two cases: We separate two cases: We separate two cases: We separate two cases: We separate two cases:

$(\sigma' = \phi \in \mathcal{T}_{\rm s})$: By induction on the structure of derivations. By induction on the structure of derivations. By induction on the structure of derivations. By induction on the structure of derivations.

(Ax): Then $\mathcal{D} = \langle Ax \rangle :: \Gamma, x{:}\rho \vdash_{\rm E} x : \sigma$, with $\rho \leq_{\rm E} \sigma$. Since $\Gamma' \leq_{\rm E} \Gamma, x{:}\rho$, there exists $x{:}\rho' \in \Gamma'$ such that $\rho' \leq_{\rm E} \rho \leq_{\rm E} \sigma \leq_{\rm E} \sigma'$. Take $\mathcal{D}' = \langle Ax \rangle :: \Gamma' \vdash_{\rm E} x : \sigma'$, then $\mathcal{D} \preceq \mathcal{D}'$.

$(\cap I)$: By induction.

$(\to I)$: Then $\sigma = \rho \to \psi$, and $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma, x{:}\rho \vdash_{\rm E} M' : \psi, \to I \rangle :: \Gamma \vdash_{\rm E} \lambda x.M' : \rho \to \psi$. Since $\sigma' \in \mathcal{T}_{\rm s}$, $\sigma' = \rho' \to \psi'$ such that $\rho' \leq_{\rm E} \rho$ and $\psi \leq_{\rm E} \psi'$. Then $\Gamma', x{:}\rho' \leq_{\rm E} \Gamma, x{:}\rho$, and by induction, there exists $\mathcal{D}'_1 \succeq \mathcal{D}_1$ such that $\mathcal{D}'_1 :: \Gamma', x{:}\rho' \vdash_{\rm E} M' : \psi'$. Now take $\mathcal{D}' = \langle \mathcal{D}'_1 :: \Gamma', x{:}\rho' \vdash_{\rm E} M' : \psi', \to I \rangle :: \Gamma' \vdash_{\rm E} \lambda x.M' : \rho' \to \psi'$, then $\mathcal{D} \preceq \mathcal{D}'$.

$(\to E)$: Then $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma \vdash_{\rm E} M_1 : \gamma \to \phi, \mathcal{D}_2 :: \Gamma \vdash_{\rm E} M_2 : \gamma, \to E \rangle :: \Gamma \vdash_{\rm E} M_1 M_2 : \phi$, so $\sigma = \phi$; without loss of generality, let $\sigma' = \phi'$. Since $\gamma \to \phi \leq_{\rm E} \gamma \to \phi'$, we have $\mathcal{D}'_1 :: \Gamma' \vdash_{\rm E} M'_1 : \gamma \to \phi'$, by induction, such that $\mathcal{D}'_1 \succeq \mathcal{D}_1$; notice that $\mathcal{D}_2 \preceq \mathcal{D}_2$. Take $\mathcal{D}' = \langle \mathcal{D}'_1, \mathcal{D}_2, \to E \rangle$, then $\mathcal{D} \preceq \mathcal{D}'$.

$(\sigma' = \sigma'_1 \cap \cdots \cap \sigma'_n)$: By Lemma 4.2, for $i \in \underline{n}$, $\sigma \leq_{\rm E} \phi'_i \in \mathcal{T}_{\rm s}$; by part (i), there exists $\mathcal{D}'_i \succeq \mathcal{D}_i$ such that $\mathcal{D}'_i :: \Gamma' \vdash_{\rm E} M : \phi'_i$. Take $\mathcal{D}' = \langle \mathcal{D}'_i, \ldots, \mathcal{D}'_n, \cap I \rangle$, then $\mathcal{D} \preceq \mathcal{D}'$.

ii) Easy, from Definition 9.2. ■

Notice that the first part of this proof is constructive, and that, therefore, we can even define a mapping that produces *one* particular larger derivation.

We should perhaps point out that the contra-variance of '$\preceq$' on derivations is not used in this lemma: in step $(\to E)$, the derivation for M_2 is not changed at all. However, this is not the issue: we only need to show that *there exists* a larger derivation that derives the required judgement. The contra-variance of the relation '$\preceq$' is needed in the proof of Lemma 9.12.

9.2 Derivation reduction

In this section, we will define a notion of reduction on derivations $\mathcal{D} :: \Gamma \vdash_{\text{E}} M:\sigma$. This will follow ordinary reduction, by contracting typed redexes that occur in $\mathcal{D}$, i.e. redexes for sub-terms of M of the shape $(\lambda x.P)Q$, for which we have a subderivation like:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma,x{:}\sigma \vdash P:\phi\end{array}}{\Gamma \vdash \lambda x.P:\sigma\to\phi}\,(\to I) \qquad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash Q:\sigma\end{array}}{\Gamma \vdash (\lambda x.P)Q:\phi}\,(\to E)$$

The effect of this reduction will be that the derivation for the redex $(\lambda x.P)Q$ will be replaced by a derivation for the contractum; this can be regarded as a generalisation of cut-elimination, but has, because the system at hand uses intersection types together with the relation '$\leq_{\text{E}}$', to be defined with care. Contracting a derivation for a redex

$$\dfrac{\dfrac{\begin{array}{c}\dfrac{}{\Gamma,x{:}\sigma \vdash x:\psi}\,(\sigma \leq_{\text{E}} \psi)\\ \boxed{\mathcal{D}_1}\\ \Gamma,x{:}\sigma \vdash P:\phi\end{array}}{\Gamma \vdash \lambda x.P:\sigma\to\phi}\,(\to I) \qquad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash Q:\sigma\end{array}}{\Gamma \vdash (\lambda x.P)Q:\phi}\,(\to E) \qquad \text{naively gives} \qquad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \dfrac{\Gamma \vdash Q:\sigma}{\Gamma \vdash Q:\psi}\,(\sigma \leq_{\text{E}} \psi)\\ \boxed{\mathcal{D}_1}\\ \Gamma \vdash P[Q/x]:\phi\end{array}$$

but this is not a correct derivation. The $(\leq_{\text{E}})$-step 'to be applied at the end of $\mathcal{D}_2$' has to be 'pushed upwards', resulting in:

$$\begin{array}{c}\boxed{\mathcal{D}_2'}\\ \Gamma \vdash Q:\psi\\ \boxed{\mathcal{D}_1}\\ \Gamma \vdash P[Q/x]:\phi\end{array}$$

(this is possible since $\mathcal{D}_2'$ exists because of Lemma 9.3, and that then $\mathcal{D}_2 \preceq \mathcal{D}_2'$). This, in general, changes the structure of the derivation, making an inductive reasoning more complicated.

Reduction on derivations is formally defined by first defining substitution on derivations:

Definition 9.4 (Derivation substitution) For the derivations $\mathcal{D}_0 :: \Gamma \vdash_{\text{E}} N:\sigma$ and $\mathcal{D} :: \Gamma,x{:}\sigma \vdash_{\text{E}} M:\tau$, the derivation

$$\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\text{E}} M[N/x]:\tau,$$

the result of substituting $\mathcal{D}_0$ for $x{:}\sigma$ in $\mathcal{D}$, is inductively defined by:

i) $\mathcal{D} = \langle Ax \rangle :: \Gamma,x{:}\sigma \vdash_{\text{E}} x:\psi$, with $\sigma \leq_{\text{E}} \psi$. Let $\mathcal{D}_0'$ be such that $\mathcal{D}_0 \preceq \mathcal{D}_0' :: \Gamma \vdash_{\text{E}} N:\psi$, then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \mathcal{D}_0'$.

ii) $\mathcal{D} = \langle Ax\rangle :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} y:\psi$; then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle Ax\rangle :: \Gamma \vdash_{\mathrm{E}} y:\psi$.

iii) $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_n,\cap I\rangle :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} M:\tau_1\cap\cdots\cap\tau_n$, so for $i \in \underline{n}$, $\mathcal{D}_i :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} M:\psi_i$. Let

$$\mathcal{D}'_i = \mathcal{D}_i\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\psi_i,$$

then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle \mathcal{D}'_1,\ldots,D'_n,\cap I\rangle :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\tau_1\cap\cdots\cap\tau_n$.

iv) $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma,x{:}\sigma,y{:}\rho \vdash_{\mathrm{E}} M_1:\psi,\to I\rangle :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} \lambda y.M_1:\rho\to\psi$. Let

$$\mathcal{D}'_1 = \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma,y{:}\rho \vdash_{\mathrm{E}} M_1[N/x]:\psi$$

Then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle \mathcal{D}'_1,\to I\rangle :: \Gamma \vdash_{\mathrm{E}} (\lambda y.M_1)[N/x]:\rho\to\psi$.

v) $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} P:\rho\to\psi, \mathcal{D}_2 :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} Q:\rho,\to E\rangle :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} PQ:\psi$. Let

$$\begin{array}{rcl} \mathcal{D}'_1 &=& \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\mathrm{E}} P[N/x]:\rho\to\psi, \text{and} \\ \mathcal{D}'_2 &=& \mathcal{D}_2\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\mathrm{E}} Q[N/x]:\rho, \end{array}$$

then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle \mathcal{D}'_1,\mathcal{D}'_2,\to E\rangle :: \Gamma \vdash_{\mathrm{E}} (PQ)[N/x]:\psi$.

Notice that, in part (Ax), we do not specify which $\mathcal{D}'_0$ to take. Lemma 9.3 guarantees its existence, not its uniqueness; however, by the remark following that lemma, we can assume $\mathcal{D}'_0$ to be unique. Moreover, by part (*iii*),

$$(\langle \cap I\rangle :: \Gamma,x{:}\sigma \vdash_{\mathrm{E}} M:\omega)\ [\mathcal{D}_0/x{:}\sigma] \ = \ \langle \cap I\rangle :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\omega.$$

Before coming to the definition of derivation-reduction, we need to define the concept of 'position of a subderivation in a derivation.'

Definition 9.5 Let $\mathcal{D}$ be a derivation, and $\mathcal{D}'$ be a subderivation of $\mathcal{D}$. The position p of $\mathcal{D}'$ in $\mathcal{D}$ is defined by:

i) If $\mathcal{D}' = \mathcal{D}$, then $p = \varepsilon$.

ii) If the position of $\mathcal{D}'$ in $\mathcal{D}_1$ is q, and $\mathcal{D} = \langle \mathcal{D}_1,\to I\rangle$, or $\mathcal{D} = \langle \mathcal{D}_1,\mathcal{D}_2,\to E\rangle$, then $p = 1q$.

iii) If the position of $\mathcal{D}'$ in $\mathcal{D}_2$ is q, and $\mathcal{D} = \langle \mathcal{D}_1,\mathcal{D}_2,\to E\rangle$, then $p = 2q$.

iv) If the position of $\mathcal{D}'$ in $\mathcal{D}_i$ ($i \in \underline{n}$) is q, and $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_n,\cap I\rangle$, then $p = q$.

We now can give a clear definition of reductions on derivations; notice that this reduction corresponds to contracting a redex $(\lambda x.M)N$ in the term involved only if that redex appears in the derivation in a subderivation with type different from ω.

Definition 9.6 We define the notion $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\sigma$ *reduces to* $\mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} M':\sigma$ *at position* p *with redex* R by:

i) $\sigma = \phi \in \mathcal{T}_{\mathrm{s}}$.

a) $\mathcal{D} = \langle\langle \mathcal{D}_1,\to I\rangle,\mathcal{D}_2,\to E\rangle :: \Gamma \vdash_{\mathrm{E}} (\lambda x.M)N:\phi$. Then $\mathcal{D}$ is shaped like:

$$\dfrac{\dfrac{\begin{array}{c}\boxed{\mathcal{D}_1}\\ \Gamma,x{:}\tau \vdash M:\phi\end{array}}{\Gamma \vdash \lambda x.M:\tau\to\phi} \qquad \begin{array}{c}\boxed{\mathcal{D}_2}\\ \Gamma \vdash N:\tau\end{array}}{\Gamma \vdash (\lambda x.M)N:\phi}$$

Then $\mathcal{D}$ reduces to $\mathcal{D}_1\,[\mathcal{D}_2/x{:}\tau] :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\phi$ at position ε with redex $(\lambda x.M)N$.

b) If $\mathcal{D}_1$ reduces to $\mathcal{D}'_1$ at position p with redex R, then

1) $\mathcal{D} = \langle \mathcal{D}_1,\to I\rangle :: \Gamma \vdash_{\mathrm{E}} \lambda x.P:\rho\to\phi$ reduces at position $1p$ with redex R to $\mathcal{D}' = \langle \mathcal{D}'_1,\to I\rangle :: \Gamma \vdash_{\mathrm{E}} \lambda x.P':\rho\to\phi$.

2) $\mathcal{D} = \langle \mathcal{D}_1,\mathcal{D}_2,\to E\rangle :: \Gamma \vdash_{\mathrm{E}} PQ:\phi$ reduces to $\mathcal{D}' = \langle \mathcal{D}'_1,\mathcal{D}_2,\to E\rangle :: \Gamma \vdash_{\mathrm{E}} P'Q:\phi$ at position $1p$ with redex R.

3) $\mathcal{D} = \langle \mathcal{D}_2, \mathcal{D}_1, \to E\rangle :: \Gamma \vdash_{\mathrm{E}} PQ:\phi$ reduces to $\mathcal{D}' = \langle \mathcal{D}_2, \mathcal{D}'_1, \to E\rangle :: \Gamma \vdash_{\mathrm{E}} PQ':\phi$ at position $2p$ with redex R.

ii) $\sigma = \phi_1 \cap \cdots \cap \phi_n$. If $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\phi_1 \cap \cdots \cap \phi_n$, then, for every $i \in \underline{n}$, there is a $\mathcal{D}_i$, such that $\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i$, and $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I\rangle$. If there is an $i \in \underline{n}$ such that $\mathcal{D}_i$ reduces to $\mathcal{D}'_i$ at position p with redex $\mathrm{R} = (\lambda x.P)Q$ (a subterm of M), then, for all $1 \leq j \neq i \leq n$, either:

a) there is no redex at position p in $\mathcal{D}_j$ because there is no subderivation at that position because the position is surrounded by a subterm that is typed with ω, and $\mathcal{D}'_j = \mathcal{D}_j$, with $P[Q/x]$ instead of $(\lambda x.P)Q$, or

b) there exists $\mathcal{D}'_j$ such that $\mathcal{D}_j$ reduces to $\mathcal{D}'_j$ at position p with redex R.

Then $\mathcal{D}$ reduces to $\langle \mathcal{D}'_1, \ldots, \mathcal{D}'_n, \cap I\rangle$ at position p with redex R.

iii) We write $\mathcal{D} \to_{\mathcal{D}} \mathcal{D}'$ if there exists a position p and redex R such that $\mathcal{D}$ reduces to $\mathcal{D}'$ at position p with redex R. If $\mathcal{D}_1 \to_{\mathcal{D}} \mathcal{D}_2 \to_{\mathcal{D}} \mathcal{D}_3$, then $\mathcal{D}_1 \to_{\mathcal{D}} \mathcal{D}_3$.

We abbreviate '$\mathcal{D}$ is strongly normalisable with respect to $\to_{\mathcal{D}}$' by '$SN(\mathcal{D})$', and use $\mathcal{SN}$ for the set of strongly normalisable derivations: $\mathcal{SN} = \{\mathcal{D} \mid SN(\mathcal{D})\}$.

Notice that the transformation needed as suggested in the beginning of this section is performed by the substitution operation on derivations, in part (*i.a*)). Also, for example,

$$\langle \cap I\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} (\lambda x.M)N:\omega \;\to_{\mathcal{D}}\; \langle \cap I\rangle :: \Gamma \vdash_{\mathrm{E}} M[N/x]:\omega.$$

at position ε with redex $(\lambda x.M)N$. Also, remark that, if $\Gamma \vdash_{\mathrm{E}} (\lambda x.M)N:\phi$, then neither $\lambda x.M$ nor M are typed with ω, so part (*ii*) is well defined.

The following lemma states the relation between derivation reduction and β-reduction.

Lemma 9.7 *Let $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\sigma$, and $\mathcal{D} \to_{\mathcal{D}} \mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} N:\sigma$, then $M \twoheadrightarrow_{\beta} N$.*

Proof: Implied by the above definition. ■

The following states some standard properties of strong normalisation.

Lemma 9.8 *i) $SN(\langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle) \Rightarrow SN(\mathcal{D}_1)$ & $SN(\mathcal{D}_2)$.*

ii) $SN(\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} xM_1 \cdots M_n:\sigma\to\phi)$ & $SN(\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} N:\sigma) \Rightarrow SN(\langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle)$.

iii) $SN(\langle \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_n, \cap I\rangle)$ if and only if, for all $i \in \underline{n}$, $SN(\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i)$.

iv) If $SN(\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} \mathrm{C}[M[N/x]]:\sigma)$, and $SN(\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} N:\rho)$, then there exists a derivation $\mathcal{D}_3$ such that $SN(\mathcal{D}_3 :: \Gamma \vdash_{\mathrm{E}} \mathrm{C}[(\lambda y.M)N]:\sigma)$.

Proof: Standard, using Definition 9.6. ■

Example 9.9 Let

$$\begin{array}{l} \mathcal{D}_1 :: \;\vdash_{\mathrm{E}} \Theta:((\alpha\to\beta\to\gamma)\cap\alpha)\to((\gamma\to\delta)\cap\beta)\to\delta, \\ \mathcal{D}_2 :: \;\vdash_{\mathrm{E}} \Theta:\tau\to(\omega\to\phi)\to\phi, \text{and} \\ \mathcal{D} \;\, :: \;\vdash_{\mathrm{E}} \Theta\Theta:(\omega\to\phi)\to\phi \end{array}$$

be the derivations from Example 6.18, and let $\mathcal{D}'_2$ be the subderivation

$$\cfrac{\cfrac{\overline{x{:}\tau, y{:}\omega\to\phi \vdash y:\omega\to\phi} \qquad \overline{x{:}\tau, y{:}\omega\to\phi \vdash xxy:\omega}\;(\cap I)}{x{:}\tau, y{:}\omega\to\phi \vdash y(xxy):\phi}\;(\to E)}{x{:}\tau \vdash \lambda y.y(xxy):(\omega\to\phi)\to\phi}\;(\to I)$$

that occurs in $\mathcal{D}_2$. Contracting $\mathcal{D}$ gives $\mathcal{D}'_2[\mathcal{D}_1/x{:}\tau]$, i.e.:

$$\dfrac{\dfrac{\dfrac{}{y{:}\omega\to\rho \vdash y:\omega\to\phi} \qquad \dfrac{}{y{:}\omega\to\rho \vdash (\Theta\Theta y):\omega}\,(\cap I)}{y{:}\omega\to\rho \vdash y(\Theta\Theta y):\phi}\,(\to E)}{\vdash \lambda y.y(\Theta\Theta y):(\omega\to\phi)\to\phi}\,(\to I)$$

Notice that this last derivation is in normal form, although the term $\lambda y.y(\Theta\Theta y)$ is not.

The following lemma is needed in the proof of Theorem 9.17.

Lemma 9.10 If $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\sigma$, with $\mathcal{D}$ in normal form, then there exists $A \in \mathcal{A}$ such that $A \sqsubseteq M$ and $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} A:\sigma$.

Proof: By induction on the structure of derivations.

($\mathcal{D} = \langle Ax \rangle$): Immediate.

($\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_n,\cap I \rangle$): Then $\sigma = \phi_1\cap\cdots\cap\phi_n$ and, for every $i \in \underline{n}$, $\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i$, and, by induction there exists $A_i \in \mathcal{A}$ such that $A_i \sqsubseteq M$ and $\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} A_i:\phi_i$. Notice that then these A_i are compatible, so $\sqcup_n A_i$ exists; since each $A_j \sqsubseteq \sqcup_n A_i$, by Lemma 6.1 also $\mathcal{D}_j :: \Gamma \vdash_{\mathrm{E}} \sqcup_n A_i:\phi_j$ for all $j \in \underline{n}$. Then, by rule $(\cap I)$, also $\Gamma \vdash_{\mathrm{E}} \sqcup_n A_i:\phi_1\cap\cdots\cap\phi_n$. Notice that, by Lemma 2.10, $\sqcup_n A_i \sqsubseteq M$.

($\mathcal{D} = \langle \mathcal{D}_1,\to I \rangle$): Then $M \equiv \lambda x.M'$, and $\sigma = \rho\to\phi$, and $\Gamma,x{:}\rho \vdash_{\mathrm{E}} M':\phi$. So, by induction, there exists $A' \in \mathcal{A}$ such that $A' \sqsubseteq M'$ and $\Gamma,x{:}\rho \vdash_{\mathrm{E}} A':\phi$. Then, by rule $(\to I)$ we obtain $\Gamma \vdash_{\mathrm{E}} \lambda x.A':\rho\to\phi$. Notice that $\lambda x.A' \sqsubseteq \lambda x.M'$; since ϕ is strict, A' is not $\bot$, so $\lambda x.A' \in \mathcal{A}$.

($\mathcal{D} = \langle \mathcal{D}_1,\mathcal{D}_2,\to E \rangle$): Then $M \equiv M_1M_2$, $\sigma = \phi$, and there is a τ such that both $\Gamma \vdash_{\mathrm{E}} M_1:\tau\to\phi$, and $\Gamma \vdash_{\mathrm{E}} M_2:\tau$. Then, by induction, there are $A_1,A_2 \in \mathcal{A}$ such that $A_1 \sqsubseteq M_1$, $A_2 \sqsubseteq M_2$, $\Gamma \vdash_{\mathrm{E}} A_1:\tau\to\phi$, and $\Gamma \vdash_{\mathrm{E}} A_2:\tau$. Then, by $(\to E)$, $\Gamma \vdash_{\mathrm{E}} A_1A_2:\phi$. Notice that $A_1A_2 \sqsubseteq M_1M_2$. Since $\mathcal{D}$ is in normal form, $\mathcal{D}_1$ does not finish with $(\to I)$, so A_1 is not an abstraction. Since $\tau\to\phi$ is strict, neither can it be $\bot$; then $A_1A_2 \in \mathcal{A}$. ■

Notice that the case $\sigma = \omega$ is present in the case $(\cap I)$ of the proof. Then $n = 0$, and $\sqcup_n A_i = \bot$. Moreover, since A need not be the same as M, the constructed derivation for it is not exactly the same, since it deals with a different term; however, it has the same structure in terms of applied derivation rules, which we defined above as being equal.

9.3 Strong normalisation result

In this section, we will come to the proof of a strong normalisation result for derivation reduction. In line with the other results shown above, in order to show that each derivation is strongly normalisable with respect to '$\to_{\mathcal{D}}$', a notion of computable derivations is introduced. We will show that all computable derivations are strongly normalisable with respect to derivation reduction, and then that all derivations in '$\vdash_{\mathrm{E}}$' are computable.

Definition 9.11 The Computability Predicate *Comp* $(\mathcal{D})$ is defined inductively on types by:

$$\begin{array}{lcl}
\mathit{Comp}\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\varphi) & \Longleftrightarrow & \mathit{SN}(\mathcal{D}) \\
\mathit{Comp}\,(\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} M:\sigma\to\phi) & \Longleftrightarrow & \\
\multicolumn{3}{r}{(\ \mathit{Comp}\,(\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} N:\sigma) \Rightarrow \mathit{Comp}\,(\langle \mathcal{D}_1,\mathcal{D}_2,\to E \rangle :: \Gamma \vdash_{\mathrm{E}} MN:\phi)\ )} \\
\mathit{Comp}\,(\langle \mathcal{D}_1,\ldots,\mathcal{D}_n,\cap I \rangle :: \Gamma \vdash_{\mathrm{E}} M:\phi_1\cap\cdots\cap\phi_n) & \Longleftrightarrow & \forall i \in \underline{n}\ [\mathit{Comp}\,(\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i)]
\end{array}$$

Notice that, as a special case for the third rule, we get *Comp* $(\langle \cap I \rangle :: \Gamma \vdash_{\mathrm{E}} M:\omega)$.

The following lemma formulates the relation between the computability predicate and the relation '$\preceq$' on derivations, and is crucial for the proof of Theorem 9.15. The main difference between the solution of [10] and the one presented here lies in the fact that here we need to prove this lemma, whereas in [10] – where rule $(\leq_{\mathrm{S}})$ corresponds to $(\cap E)$ which behaviour is already captured in the definition of *Comp* – it is not needed at all (see Section 9.5).

Lemma 9.12 *If* $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\sigma)$, *and* $\mathcal{D} \preceq \mathcal{D}'$, *then* $Comp\,(\mathcal{D}')$.

Proof: By induction on the structure of types. Notice that, by Lemma 9.3, $\mathcal{D}' = \Gamma' \vdash_{\mathrm{E}} M{:}\sigma'$, with $\Gamma' \leq_{\mathrm{E}} \Gamma$, $\sigma \leq_{\mathrm{E}} \sigma'$. We distinguish two cases:

$(\sigma' = \psi \in \mathcal{T}_{\mathrm{s}})$: $(\sigma = \varphi)$: Since $\varphi \leq_{\mathrm{E}} \psi$, also $\psi = \varphi$, and the result is immediate.

$(\sigma = \rho{\rightarrow}\phi)$: Then $\psi = \rho'{\rightarrow}\phi'$, with $\rho' \leq_{\mathrm{E}} \rho, \phi \leq_{\mathrm{E}} \phi'$, and let $\mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} M{:}\rho'{\rightarrow}\phi'$, which exists by Lemma 4.5. To show $Comp\,(\mathcal{D}')$, following Definition 9.11, we assume $Comp\,(\mathcal{D}'_0 :: \Gamma \vdash_{\mathrm{E}} N{:}\rho')$, and use this to show that $\langle \mathcal{D}',\mathcal{D}'_0,{\rightarrow}E\rangle :: \Gamma \vdash_{\mathrm{E}} MN{:}\phi'$. Let $\mathcal{D}_0$ be such that $\mathcal{D}'_0 \preceq \mathcal{D}_0 :: \Gamma \vdash_{\mathrm{E}} N{:}\rho$ (which, again, exists by Lemma 4.5), we get $Comp\,(\mathcal{D}_0)$ by induction from $Comp\,(\mathcal{D}'_0)$. Assuming $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\rho{\rightarrow}\phi)$, by Definition 9.11, $Comp\,(\langle \mathcal{D},\mathcal{D}_0,{\rightarrow}E\rangle :: \Gamma \vdash_{\mathrm{E}} MN{:}\phi)$. Since

$$\langle \mathcal{D},\mathcal{D}_0,{\rightarrow}E\rangle \preceq \langle \mathcal{D}',\mathcal{D}'_0,{\rightarrow}E\rangle :: \Gamma \vdash_{\mathrm{E}} MN{:}\phi',$$

we get, by induction $Comp\,(\langle \mathcal{D}',\mathcal{D}'_0,{\rightarrow}E\rangle)$. So $Comp\,(\mathcal{D}')$ by Definition 9.11.

$(\sigma = \phi_1{\cap}\cdots{\cap}\phi_n)$: If $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\phi_1{\cap}\cdots{\cap}\phi_n)$, then $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_n,{\cap}I\rangle$, by Definition 9.11, and $Comp\,(\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M{:}\phi_i)$ for $i \in \underline{n}$. Since $\phi_1{\cap}\cdots{\cap}\phi_n \leq_{\mathrm{E}} \psi$, by Lemma 4.2, there exists $i \in \underline{n}$ such that $\phi_i \leq_{\mathrm{E}} \psi$. Then $\mathcal{D} \preceq \mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M{:}\psi$ and, by induction, $Comp\,(\mathcal{D}_i)$.

$(\sigma' = \sigma'_1{\cap}\cdots{\cap}\sigma'_n)$: Since $\sigma \leq_{\mathrm{E}} \sigma'_1{\cap}\cdots{\cap}\sigma'_n$, by Lemma 4.2, $\sigma = \sigma_1{\cap}\cdots{\cap}\sigma_m$, and for all $i \in \underline{n}$ there exists $j \in \underline{m}$ such that $\phi_j \leq_{\mathrm{E}} \phi'_i$. If $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\sigma_1{\cap}\cdots{\cap}\sigma_m)$, then, by Definition 9.11, for every $j \in \underline{m}$ there exist $\mathcal{D}_j$ such that $Comp\,(\mathcal{D}_j :: \Gamma \vdash_{\mathrm{E}} M{:}\phi_j)$, and $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_m,{\cap}I\rangle$. Let $\mathcal{D}' = \langle \mathcal{D}'_1,\ldots,\mathcal{D}'_n,{\cap}I\rangle$; since $\mathcal{D} \preceq \mathcal{D}'$, for every $i \in \underline{n}$ there exists $j \in \underline{m}$ such that $\mathcal{D}_j \preceq \mathcal{D}'_i :: \Gamma \vdash_{\mathrm{E}} M{:}\phi'_i$; then, by induction, $Comp\,(\mathcal{D}'_i)$ for all $i \in \underline{n}$, and, by Definition 9.11, $Comp\,(\mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} M{:}\cap_n\phi'_i)$ follows. ■

Notice that the contra-variance of the relation '$\preceq$' on derivations plays a role in the second part of this proof.

We will now prove that *Comp* satisfies the standard properties of computability predicates, being that computability implies strong normalisation, and that, for the so-called *neutral* objects, also the converse holds.

Lemma 9.13 *i)* $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\sigma) \Rightarrow SN(\mathcal{D})$.
ii) $SN(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} xM_1\cdots M_m{:}\sigma) \Rightarrow Comp\,(\mathcal{D})$.

Proof: By simultaneous induction on the structure of types.

$(\sigma = \varphi)$: Directly by Definition 9.11.

$(\sigma = \rho{\rightarrow}\phi)$: i) Let x be a variable not appearing in M, and let $\mathcal{D}' :: x{:}\rho \vdash_{\mathrm{E}} x{:}\rho$, then, by induction (*ii*), $Comp\,(\mathcal{D}')$. Assume, without loss of generality, that $x{:}\rho \in \Gamma$. From assuming $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\rho{\rightarrow}\phi)$, $Comp\,(\langle \mathcal{D},\mathcal{D}',{\rightarrow}E\rangle :: \Gamma \vdash_{\mathrm{E}} Mx{:}\phi)$ follows by Definition 9.11. Then, by induction (*i*), $SN(\langle \mathcal{D},\mathcal{D}',{\rightarrow}E\rangle)$, so also $SN(\mathcal{D})$.

ii) $Comp\,(\mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} N{:}\rho)$ gives, by induction (*i*), $SN(\mathcal{D}')$, and, by Lemma 9.8(*ii*), $SN(\langle \mathcal{D},\mathcal{D}',{\rightarrow}E\rangle :: \Gamma \vdash_{\mathrm{E}} xM_1\cdots M_mN{:}\phi)$. Then $Comp\,(\langle \mathcal{D},\mathcal{D}',{\rightarrow}E\rangle)$ by induction (*ii*), so by Definition 9.11, $Comp\,(\mathcal{D})$.

$(\sigma = \phi_1{\cap}\cdots{\cap}\phi_n)$: Easy, using Definition 9.11, Lemma 9.8(*iii*), and induction. ■

The following theorem (9.15) shows that, if the instances of rule (Ax) are to be replaced by computable derivations, then the result itself will be computable. Before coming to this result, an auxiliary lemma is proven, showing that the computability predicate is closed for subject-expansion.

Lemma 9.14 *If* $Comp\,(\mathcal{D}[\mathcal{D}'/y{:}\mu] :: \Gamma \vdash_{\rm E} M[Q/y]\overrightarrow{P}{:}\,\sigma)$ *and* $Comp\,(\mathcal{D}' :: \Gamma \vdash_{\rm E} Q{:}\,\mu)$*, then there exists a derivation* $\mathcal{D}''$ *such that* $Comp\,(\mathcal{D}'' :: \Gamma \vdash_{\rm E} (\lambda y.M)Q\overrightarrow{P}{:}\,\sigma)$*.*

Proof: By induction on the structure of types.

$$\begin{array}{lll}
(\sigma = \varphi): & Comp\,(\mathcal{D}[\mathcal{D}'/y{:}\mu] :: \Gamma \vdash_{\rm E} M[Q/y]\overrightarrow{P}{:}\,\varphi) \;\&\; Comp\,(\mathcal{D}' :: \Gamma \vdash_{\rm E} Q{:}\,\mu) & \Rightarrow (9.13(i)) \\
& SN(\mathcal{D}[\mathcal{D}'/y{:}\mu]) \;\&\; SN(\mathcal{D}') & \Rightarrow (9.8(iv)) \\
& \exists\,\mathcal{D}''\;[\,SN(\mathcal{D}'' :: \Gamma \vdash_{\rm E} (\lambda y.M)Q\overrightarrow{P}{:}\,\varphi)\,] & \Rightarrow (9.11) \\
& \exists\,\mathcal{D}''\;[\,Comp\,(\mathcal{D}'' :: \Gamma \vdash_{\rm E} (\lambda y.M)Q\overrightarrow{P}{:}\,\varphi)\,].
\end{array}$$

$$\begin{array}{ll}
(\sigma = \rho{\to}\phi):\; Comp\,(\mathcal{D}[\mathcal{D}'/y{:}\mu] :: \Gamma \vdash_{\rm E} M[Q/y]\overrightarrow{P}{:}\,\rho{\to}\phi) \;\& & \\
\qquad\qquad Comp\,(\mathcal{D}_1 :: \Gamma \vdash_{\rm E} N{:}\,\rho) \;\&\; Comp\,(\mathcal{D}' :: \Gamma \vdash_{\rm E} Q{:}\,\mu) & \Rightarrow (9.11) \\
\quad Comp\,(\langle \mathcal{D}[\mathcal{D}'/y{:}\mu],\mathcal{D}_1,{\to}E\rangle :: \Gamma \vdash_{\rm E} M[Q/y]\overrightarrow{P}N{:}\,\phi) \;\&\; Comp\,(\mathcal{D}' :: \Gamma \vdash_{\rm E} Q{:}\,\mu) & \Rightarrow (IH) \\
\quad \exists\,\mathcal{D}''\;[\,Comp\,(\langle \mathcal{D}'',\mathcal{D}_1,{\to}E\rangle :: \Gamma \vdash_{\rm E} (\lambda y.M)Q\overrightarrow{P}N{:}\,\phi)\,] & \Rightarrow (9.11) \\
\quad \exists\,\mathcal{D}''\;[\,Comp\,(\mathcal{D}'' :: \Gamma \vdash_{\rm E} (\lambda y.M)Q\overrightarrow{P}{:}\,\rho{\to}\phi)\,] &
\end{array}$$

$(\sigma = \phi_1{\cap}\cdots{\cap}\phi_n)$: By induction and Definition 9.11. ■

We now come to the Replacement Theorem.

Theorem 9.15 *Let* $\Gamma = x_1{:}\mu_1,\ldots,x_n{:}\mu_n$*,* $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\,\sigma$*, and, for every* $i \in \underline{n}$*, there are* $\mathcal{D}^i, N_i$ *such that* $Comp\,(\mathcal{D}^i :: \Gamma' \vdash_{\rm E} N_i{:}\,\mu_i)$*. Then*

$$Comp\,(\mathcal{D}[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] :: \Gamma' \vdash_{\rm E} M[\overrightarrow{N_i/x_i}]{:}\,\sigma).$$

Proof: By induction on the structure of derivations.

(Ax): Then $M \equiv x_i$, for some $i \in \underline{n}$, with $\mu_i \leq_{\rm E} \sigma$. By Lemma 4.5, $\mathcal{D}' :: \Gamma' \vdash_{\rm E} N_i{:}\,\sigma$ exists, and, by Lemma 9.3, $\mathcal{D}^i \preceq \mathcal{D}'$, so, from $Comp\,(\mathcal{D}^i)$, by Lemma 9.12, $Comp\,(\mathcal{D}')$. Notice that $\mathcal{D}' = (\langle Ax\rangle :: \Gamma \vdash_{\rm E} x{:}\,\sigma)[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}]$.

$(\cap I)$: Then $\sigma = \sigma_1{\cap}\cdots{\cap}\sigma_m$, and, for $j \in \underline{m}$, there exists $\mathcal{D}_j$, such that $\mathcal{D}_j :: \Gamma \vdash_{\rm E} M{:}\,\phi_j$ and $\mathcal{D} = \langle \mathcal{D}_1,\ldots,\mathcal{D}_m,{\cap}I\rangle$. Let, for $j \in \underline{m}$, $\mathcal{D}'_j = \mathcal{D}_j[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}] :: \Gamma \vdash_{\rm E} M[\overrightarrow{N_i/x_i}]{:}\,\phi_j$, then, by induction, $Comp\,(\mathcal{D}'_j)$. Let $\mathcal{D}' = \langle \mathcal{D}'_1,\ldots,\mathcal{D}'_m,{\cap}I\rangle$, then, by Definition 9.11,

$$Comp\,(\mathcal{D}' :: \Gamma \vdash_{\rm E} M[\overrightarrow{N_i/x_i}]{:}\,\sigma_1{\cap}\cdots{\cap}\sigma_m),$$

and $\mathcal{D}' = \mathcal{D}[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}]$.

$({\to}I)$: Then $\sigma = \rho{\to}\psi$, and $\mathcal{D} = \langle \mathcal{D}' :: \Gamma,y{:}\rho \vdash_{\rm E} M'{:}\,\psi,{\to}I\rangle :: \Gamma \vdash_{\rm E} \lambda y.M'{:}\,\rho{\to}\psi$.

$$\begin{array}{ll}
\forall i \in \underline{n}\;[\,Comp\,(\mathcal{D}^i :: \Gamma' \vdash_{\rm E} N_i{:}\,\mu_i)\,] \;\&\; Comp\,(\mathcal{D}_2 :: \Gamma' \vdash_{\rm E} P{:}\,\rho) & \\
\qquad\qquad\qquad \&\; \mathcal{D}' :: \Gamma,y{:}\rho \vdash_{\rm E} M'{:}\,\psi & \Rightarrow (IH) \\
Comp\,(\mathcal{D}'[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i},\mathcal{D}_2/y{:}\rho] :: \Gamma' \vdash_{\rm E} M'[\overrightarrow{N_i/x_i},P/y]{:}\,\psi) & \Rightarrow (9.14) \\
Comp\,(\langle\langle \mathcal{D}'[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}],{\to}I\rangle,\mathcal{D}_2,{\to}E\rangle :: \Gamma' \vdash_{\rm E} (\lambda y.M'[\overrightarrow{N_i/x_i}])P{:}\,\psi) & \Rightarrow (9.11) \\
Comp\,(\langle \mathcal{D}'[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}],{\to}I\rangle :: \Gamma' \vdash_{\rm E} \lambda y.M'[\overrightarrow{N_i/x_i}]{:}\,\rho{\to}\psi). &
\end{array}$$

and $\mathcal{D}' = \langle \mathcal{D}'[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}],{\to}I\rangle = \mathcal{D}[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}]$.

$({\to}E)$: Then $M \equiv M_1M_2$, there are $\mathcal{D}_1,\mathcal{D}_2$, and τ such that $\mathcal{D} = \langle \mathcal{D}_1,\mathcal{D}_2,{\to}E\rangle$, $\mathcal{D}_1 :: \Gamma \vdash_{\rm E} M_1{:}\,\tau{\to}\phi$, and $\mathcal{D}_2 :: \Gamma \vdash_{\rm E} M_2{:}\,\tau$. Let

$$\begin{array}{l}
\mathcal{D}'_1 = \mathcal{D}_1[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}] :: \Gamma' \vdash_{\rm E} M_1[\overrightarrow{N_i/x_i}]{:}\,\tau{\to}\phi, \text{ and} \\
\mathcal{D}'_2 = \mathcal{D}_2[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}] :: \Gamma' \vdash_{\rm E} M_2[\overrightarrow{N_i/x_i}]{:}\,\tau,
\end{array}$$

then, by induction, $Comp\,(\mathcal{D}'_1)$, and $Comp\,(\mathcal{D}'_2)$, and by Definition 9.11,

$$Comp\,(\langle \mathcal{D}'_1,\mathcal{D}'_2,\to E\rangle :: \Gamma' \vdash_{\rm E} (M_1M_2)[\overrightarrow{N_i/x_i}] : \phi),$$

Notice that $\langle \mathcal{D}_1,\mathcal{D}_2,\to E\rangle[\overrightarrow{\mathcal{D}_i/x_i{:}\mu_i}] = \langle \mathcal{D}'_1,\mathcal{D}'_2,\to E\rangle$. ■

Using this last result, we are now able to prove a strong normalisation result for derivation reduction in '$\vdash_{\rm E}$'.

Theorem 9.16 *If $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\sigma$, then $SN(\mathcal{D})$.*

Proof: For every $x_i{:}\tau_i \in \Gamma$, there exists $\mathcal{D}_{x_i} :: x_i{:}\tau_i \vdash_{\rm E} x_i : \tau_i$ and, by Lemma 9.13(*ii*), $Comp\,(\mathcal{D}_{x_i})$. Then $Comp\,(\mathcal{D}[\overrightarrow{\mathcal{D}_{x_i}/x_i{:}\tau_i}] :: \Gamma \vdash_{\rm E} M[\overrightarrow{x_i/x_i}] : \sigma)$, by Theorem 9.15. Notice that $M[\overrightarrow{x_i/x_i}] = M$ and $\mathcal{D}[\overrightarrow{\mathcal{D}_{x_i}/x_i{:}\tau_i}] = \mathcal{D}$, and by Theorem 9.13(*i*), $SN(\mathcal{D})$. ■

9.4 New proofs of Approximation and Strong Normalisation

We will now show that the approximation result is a direct consequence of the strong normalisation result proven in Section 9.3 for derivation reduction.

Using Theorem 9.16, the approximation theorem is proven as follows:

Theorem 9.17 $\Gamma \vdash_{\rm E} M{:}\sigma \iff \exists A \in \mathcal{A}(M)\ [\Gamma \vdash_{\rm E} A{:}\sigma]$.

Proof: $(\Rightarrow)$: Let $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\sigma$, then, by Theorem 9.16, $SN(\mathcal{D})$. Let the normal form of $\mathcal{D}$ with respect to '$\to_{\mathcal{D}}$' be $\mathcal{D}_0 :: \Gamma \vdash_{\rm E} N{:}\sigma$, then by Lemma 9.7, $M \twoheadrightarrow_\beta N$, and by Lemma 9.10, there is $A \in \mathcal{A}$ such that $\mathcal{D}_0 :: \Gamma \vdash_{\rm E} A{:}\sigma$, and $A \sqsubseteq N$.

$(\Leftarrow)$: As in Theorem 6.9. ■

Of course the proof of the characterisation of (head-)normalisation (Section 6.3) does not change, since it depends only on the approximation result.

We will now show a new proof for the result that all terms typeable in the system without ω are strongly normalisable. This result is also obtained via the strong normalisation result proven above for derivation reduction.

Theorem 9.18 $\exists \Gamma,\sigma\ [\Gamma \vdash_{\not\omega} M{:}\sigma] \iff$ *M is strongly normalisable with respect to '$\to_\beta$'.*

Proof: $(\Rightarrow)$: If $\mathcal{D} :: \Gamma \vdash_{\not\omega} M{:}\sigma$, then by Lemma 8.2, also $\mathcal{D} :: \Gamma \vdash_{\rm E} M{:}\sigma$. By Theorem 9.16, $\mathcal{D}$ is strongly normalisable with respect to '$\to_{\mathcal{D}}$'. Since $\mathcal{D}$ contains no ω, all redexes in M correspond to redexes in $\mathcal{D}$. Since derivation reduction does not introduce ω, also M is strongly normalisable with respect to '$\to_\beta$'.

$(\Leftarrow)$: As in Theorem 8.15. ■

9.5 Derivation reduction in other type assignment systems

The results shown above hold of course also when restricted to either '$\vdash_{\rm S}$', or the relevant system '$\vdash_{\rm R}$' (see Section 10.1, which corresponds to the restricted system of Definition 3.6), since these are proper subsystems of '$\vdash_{\rm E}$'. However, we can also show these results directly in those other systems; we will illustrate the result of Section 9.3 by looking briefly at those systems and see that there the result comes more easily.

It is worthwhile to remark that, in fact, it is the presence of the contra-variant relation '$\leq_{\rm E}$' on types, and, especially, the derivation rule (Ax), that greatly complicates the possible solution to the main problem dealt with above. Restricting the setting to the *relevant* system '$\vdash_{\rm R}$' gives a rather straightforward solution.

We will not discuss the proof for the result in detail here, since it would be very similar to the proof that was given above, or to that in [10]. The main difference lies in the fact that a relevant system is not closed for '$\leq_{\rm E}$', so in particular no variant of Lemma 9.12 needs to be proven. That lemma is essential to prove part (Ax) of the proof of Theorem 9.15; for '$\vdash_{\mathrm{R}}$' this part would be trivial, since then the context would consist of a single statement $x{:}\mu$:

(Ax): Then $n=1$, $x{:}\mu = \Gamma'$. By assumption, $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{R}} N{:}\,\mu)$, which immediately gives $Comp\,(\mathcal{D} :: \Gamma \vdash_{\mathrm{R}} x[N/x]:\mu)$.

A direct proof that derivation reduction is strongly normalisable was given for '$\vdash_{\mathrm{S}}$' in [10]. Again we will omit almost all the proof for that result here, since it would be very similar to the proof that was given above. The difference between '$\vdash_{\mathrm{S}}$' and the one considered in this section, rule $(\cap E)$ versus (Ax), makes the first part of the Replacement Lemma become:

$(\cap E)$: Then $\mu_i = \sigma_1 \cap \cdots \cap \sigma_m$ so $x_i{:}\sigma_1 \cap \cdots \cap \sigma_m \in \Gamma'$, and $\sigma = \phi_k$ for some $k \in \underline{m}$. By assumption, $Comp\,(\mathcal{D}^i :: \Gamma \vdash_{\mathrm{S}} N_i : \sigma_1 \cap \cdots \cap \sigma_m)$, and, since $\sigma_1 \cap \cdots \cap \sigma_m$ is an intersection type, by definition of $\overrightarrow{\text{computab}}$ility this implies $\mathcal{D}^j_i :: \Gamma \vdash_{\mathrm{S}} N_i : \phi_i$, where $\mathcal{D}^j_i$ is a subderivation of $\mathcal{D}^i$. Now $\mathcal{D}^0\,[\mathcal{D}^i / x_i{:}\mu_i] = \mathcal{D}^i_j$, so computable.

From these results, as above, the characterisation of normalisation, head-normalisation, and strong normalisation can be shown.

The technique used in [3] – which required $Comp\,(\Gamma, M, \sigma)$ to imply $\Gamma \vdash M{:}\sigma$ – for the strong normalisation result would not work for '$\vdash_{\mathrm{S}}$', since it needs a contra-variant type inclusion relation, or, equivalently, needs the notion of type assignment to be closed for η-reduction; the same is true for the proof of the approximation result in [7] (Section 6.1); in particular, it is needed for Lemma 6.2(*ii*). Notice that this is not the case with the technique used in this section, which gives these results directly for '$\vdash_{\mathrm{S}}$', and does not need extensionality.

10 Principal pairs for '$\vdash_{\mathrm{E}}$'

As discussed above, there exist four intersection systems for which the principal pair property is proven: a CDV-system in [21], '$\vdash_{\mathrm{BCD}}$' in [55], '$\vdash_{\mathrm{S}}$' in [6], and '$\vdash_{\mathrm{E}}$' in [7], the proof of which we will repeat here.

As already discussed above, a proof for the principal pair property normally follows the following structure. First, for each term, a specific pair (of context and type) is identified, called the *principal pair*, that is shown to be a valid pair in terms of typeability for this term (*soundness*). Then a collection of operations is identified that is proven to be sound in the sense that they, when applied to a valid pair, return a valid pair; then it is shown that, for every term, any valid pair can be obtained by applying a specific sequence of operations to the principal pair.

In this section we will follow this scheme when giving the proof for the principal pair property of '$\vdash_{\mathrm{E}}$' will be given. For each λ-term the principal pair (of context and type) will be defined. Four operations on pairs of context and types will be defined, namely *substitution*, *expansion*, *covering*, and *lifting*, that are correct and sufficient to generate all derivable pairs for λ-terms in '$\vdash_{\mathrm{E}}$'.

The proof will start by proving the principal pair property for the relevant type assignment system '$\vdash_{\mathrm{R}}$' by showing that, if $\Gamma \vdash_{\mathrm{R}} M{:}\sigma$ and $\langle \Pi, \pi \rangle$ is the principal pair for M, then there exists a chain Ch of operations, consisting of a number of expansions, at most one covering, and at most one substitution, such that $Ch\,(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$. Using this result, the principal pair property for '$\vdash_{\mathrm{E}}$' will be proven, adding a single lifting to the chain.

In [6] (see Section 5.1), the main problem to solve was to find an operation of lifting that was able to take the special role of the rule $(\cap E)$ into account. As mentioned in Section 5.1, there this operation is defined using, in fact, the contra-variant relation '$\leq_{\mathrm{E}}$', which is not sound on all pairs, but is sound on primitive pairs. Since '$\vdash_{\mathrm{E}}$' is more liberal than '$\vdash_{\mathrm{S}}$', in the sense that '$\vdash_{\mathrm{E}}$' *is* closed for the relation '$\leq_{\mathrm{E}}$', the operation of lifting as defined in [6] is a sound operation for '$\vdash_{\mathrm{E}}$' (see Theorem 10.24), i.e. is correct for all pairs. It is then easy to show that, with just the operations as defined in [6], the principal pair property holds for '$\vdash_{\mathrm{E}}$'.

However, in this section a different proof will be presented that follows a slightly different approach. The most significant difference between proofs for the principal pair 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, or 'orthogonal'. In [55], there is an overlap between operations; for example, intersections can be introduced by expansions as well as by substitutions and rise. Also, in [6] the step from the pair $\langle \Gamma, \sigma \rangle$ to $\langle \Gamma, \omega \rangle$ can be made using a lifting as well as a substitution. The operations of expansion, covering, substitution, and lifting as defined in this section are orthogonal in that sense; no kind of operation can be mimicked by another kind of operation or sequence of operations.

The difference between the operations specified in [6] and this section lie in the fact that here the operation of substitution has been changed, in a subtle, natural, but drastic way: since ω is not considered a type constant, a substitution can no longer replace a type-variable by ω. In the papers discussed above that possibility existed and, especially in [21] and [6], caused inconvenience, since there a 'normalisation-after-substitution' was called for, explicitly defined in [21], and part of the definition of substitution in [6]. The approach of this paper will be to allow of only substitutions of strict types for type variables, and to introduce a separate operation of *covering*, that deals with the assignment of ω to sub-terms.

10.1 The relevant system

The notion of *relevant* intersection type assignment is used in [29, 7, 1], and expresses that a context can only contain those types that are actually used in a derivation, i.e. uses only types that are *relevant* to reach the judgement in the conclusion. Apart from having a separate rule that deals with ω, it corresponds to the restricted system of [22] (see Definition 3.6); for convenience in proofs, we present it here as a restricted version of '$\vdash_{\mathrm{E}}$'.

Definition 10.1 *Relevant intersection type assignment* and *relevant intersection derivations* are defined by the following natural deduction system:

$$(Ax): \frac{}{x{:}\phi \vdash x:\phi} \qquad (\rightarrow I): \frac{\Gamma, x{:}\sigma \vdash M:\phi}{\Gamma \vdash \lambda x.M:\sigma\rightarrow\phi} \quad \frac{\Gamma \vdash M:\phi}{\Gamma \vdash \lambda x.M:\omega\rightarrow\tau}\ (x \text{ not in } \Gamma)$$

$$(\rightarrow E): \frac{\Gamma_1 \vdash M:\sigma\rightarrow\phi \quad \Gamma_2 \vdash N:\sigma}{\bigcap\{\Gamma_1,\Gamma_2\} \vdash MN:\phi} \qquad (\cap I): \frac{\Gamma_1 \vdash M:\phi_1 \quad \cdots \quad \Gamma_n \vdash M:\phi_n}{\bigcap_n \Gamma_i \vdash M:\phi_1\cap\cdots\cap\phi_n}\ (n \geq 0)$$

Since the derivable statements are exactly the same to those derivable in '$\vdash_{\mathrm{R}}$', we write $\Gamma \vdash_{\mathrm{R}} M:\sigma$ also for judgements derivable in this system.

Notice that, by rule $(\cap I)$, $\emptyset \vdash_{\mathrm{R}} M:\omega$, for all terms M. Notice moreover, that this system is indeed truly *relevant* in the sense of [29]: contexts contain no more type information than that actually used in the derivation, and, therefore, in the $(\rightarrow I)$-rule, only those types *actually used* in the derivation can be abstracted. This implies that, for example, for the λ-term $(\lambda ab.a)$ types like $\psi\rightarrow\phi\rightarrow\psi$ cannot be derived, only types like $\psi\rightarrow\omega\rightarrow\psi$. Likewise, for $\lambda x.x$ types like $(\phi\cap\psi)\rightarrow\phi$ cannot be derived, only types like $\phi\rightarrow\phi$ can.

Notice that, essentially, the difference between '$\vdash_{\mathrm{R}}$' and '$\vdash_{\mathrm{S}}$' lies in going from derivation rule (Ax) to $(\cap E)$. In fact, derivation rule $(\cap E)$ is implicitly present in '$\vdash_{\mathrm{R}}$', since there the intersection of types occurring in contexts is produced using the $\bigcap$-operator on contexts. The strict system does not use this operator; instead, it allows for the selection of types from an intersection type occurring in a context, regardless of the fact if all components of that intersection type are useful for the full derivation. In this sense, '$\vdash_{\text{S}}$' is not relevant. Both '$\vdash_{\text{S}}$' and '$\vdash_{\text{E}}$' are conservative extensions of '$\vdash_{\text{R}}$', in the sense that, if $\Gamma \vdash_{\text{R}} M{:}\sigma$, then also $\Gamma \vdash_{\text{S}} M{:}\sigma$ and $\Gamma \vdash_{\text{E}} M{:}\sigma$.

In much the same way as Lemma 4.9, we can show:

Lemma 10.2 $\exists \rho\, [\Gamma_1, x{:}\rho \vdash_{\text{R}} M{:}\sigma \;\&\; \Gamma_2 \vdash_{\text{R}} N{:}\rho] \iff \cap\{\Gamma_1,\Gamma_2\} \vdash_{\text{R}} M[N/x]{:}\sigma$.

Notice that $\rho = \omega$, and then $\Gamma_2 = \emptyset$ whenever $x \notin \mathit{fv}(M)$.

As for '$\vdash_{\text{BCD}}$' (Theorem 6.12), the relation between '$\vdash_{\text{E}}$' and '$\vdash_{\text{R}}$' is formulated for terms in $\mathcal{A}$ by:

Lemma 10.3 For all $A \in \mathcal{A}$: if $\Gamma \vdash_{\text{E}} A{:}\sigma$, then there are Γ',σ' such that $\Gamma \vdash_{\text{R}} A{:}\sigma$, $\sigma' \leq_{\text{E}} \sigma$ and $\Gamma \leq_{\text{E}} \Gamma'$.

Using the same technique as in Section 6.1, the following theorem can be proven.

Theorem 10.4 $\Gamma \vdash_{\text{R}} M{:}\sigma \iff \exists A \in \mathcal{A}(M)\, [\Gamma \vdash_{\text{R}} A{:}\sigma]$.

Using this approximation result for '$\vdash_{\text{R}}$', the following becomes easy.

Theorem 10.5 *If $\Gamma \vdash_{\text{E}} M{:}\sigma$, then there are $\Gamma' \geq_{\text{E}} \Gamma$, $\sigma' \leq_{\text{E}} \sigma$ such that $\Gamma' \vdash_{\text{R}} M{:}\sigma$.*

Proof: If $\Gamma \vdash_{\text{E}} M{:}\sigma$ then, by Theorem 6.9, there is an $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\text{E}} A{:}\sigma$. By Lemma 10.3, there are Γ',σ' such that $\Gamma' \vdash_{\text{R}} M{:}\sigma'$, $\sigma' \leq_{\text{E}} \sigma$ and $\Gamma \leq_{\text{E}} \Gamma'$. Then, by Theorem 10.4, $\Gamma' \vdash_{\text{R}} M{:}\sigma'$. ■

10.2 Principal Pairs

Principal pairs for both '$\vdash_{\text{R}}$' and '$\vdash_{\text{E}}$' are defined by:

Definition 10.6 (Principal Pairs) *i*) For $A \in \mathcal{A}$, we define $\mathit{pp}(A)$, the *principal pair* of A, by:

- *a*) $\mathit{pp}(\bot) = \langle \emptyset, \omega \rangle$.
- *b*) $\mathit{pp}(x) = \langle \{x{:}\varphi\}, \varphi \rangle$.
- *c*) If $A \neq \bot$, and $\mathit{pp}(A) = \langle \Pi, \pi \rangle$, then:
  - *1*) If x occurs free in A, and $x{:}\sigma \in \Pi$, then $\mathit{pp}(\lambda x.A) = \langle \Pi \backslash x, \sigma \to \pi \rangle$.
  - *2*) Otherwise $\mathit{pp}(\lambda x.A) = \langle \Pi, \omega \to \pi \rangle$.
- *d*) If for $i \in \underline{n}$, $\mathit{pp}(A_i) = \langle \Pi_i, \pi_i \rangle$ (disjoint in pairs), then

$$\mathit{pp}(xA_1 \cdots A_n) = \langle \cap_n \Pi_i \cap \{x{:}\pi_1 \to \cdots \to \pi_n \to \varphi\}, \varphi \rangle,$$

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

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

The principal pairs in the systems as presented in [21], '$\vdash_{\text{BCD}}$' in [55], and '$\vdash_{\text{S}}$' in [6] are exactly as above. Since the essential type assignment system is a subsystem of '$\vdash_{\text{BCD}}$', and it is a super-system '$\vdash_{\text{S}}$', which, in turn, is a super-system of '$\vdash_{\text{R}}$', this is not surprising.

The following result is almost immediate:

Lemma 10.7 If $\mathit{pp}(A) = \langle \Pi, \pi \rangle$, then $\Pi \vdash_{\text{R}} A{:}\pi$.

Proof: Easy. ■

The notion of principal pairs for terms in $\mathcal{A}$ will be generalised to arbitrary λ-terms in Definition 10.29.

10.3 Substitution

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 approach of [55], this kind of operation will proven to be sufficient.

Definition 10.8 *i*) The *substitution* $(\varphi \mapsto \psi) : \mathcal{T} \to \mathcal{T}$, where φ is a type-variable and $\psi \in \mathcal{T}_{\mathrm{s}}$, is defined by:

$$\begin{array}{lcll} (\varphi \mapsto \psi)\varphi & = & \psi & \\ (\varphi \mapsto \psi)\varphi' & = & \varphi', & \textit{if } \varphi \not\equiv \varphi' \\ (\varphi \mapsto \psi)\sigma \to \phi & = & (\varphi \mapsto \psi)\sigma \to (\varphi \mapsto \psi)\phi & \\ (\varphi \mapsto \psi)\phi_1 \cap \cdots \cap \phi_n & = & (\varphi \mapsto \psi)\phi_1 \cap \cdots \cap (\varphi \mapsto \psi)\phi_n & \end{array}$$

ii) If S_1 and S_2 are substitutions, then so is $S_1 \circ S_2$, where $S_1 \circ S_2\,(\sigma) = S_1\,(S_2\,(\sigma))$.

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

iv) $S\,(\langle \Gamma,\sigma \rangle) = \langle S\,(\Gamma), S\,(\sigma) \rangle$.

The operation of substitution is sound for '$\vdash_{\mathrm{R}}$'.

Theorem 10.9 *If $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$, then for every substitution S: if $S\,(\langle \Gamma,\sigma \rangle) = \langle \Gamma',\sigma' \rangle$, then $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$.*

Proof: By straightforward induction on the definition of '$\vdash_{\mathrm{R}}$'. ■

The following is needed in the proof of Theorem 10.27.

Lemma 10.10 Let $\tau \in \mathcal{T}_{\mathrm{s}}$ and S be a substitution such that $S\,(\phi) = \phi'$. Then:

i) If $S\,(\Gamma, x{:}\sigma) = \Gamma', x{:}\sigma'$, then $S\,(\langle \Gamma, \sigma \to \phi \rangle) = \langle \Gamma', \sigma' \to \phi' \rangle$.

ii) If for every $i \in \underline{n}$, $S\,(\langle \Gamma_i,\sigma_i \rangle) = \langle \Gamma_i',\sigma_i' \rangle$, then

$$S\,(\langle \cap_n \Gamma_i \cap \{x{:}\sigma_1 \to \cdots \to \sigma_n \to \phi\}, \phi \rangle) \;=\; \langle \cap_n \Gamma_i' \cap \{x{:}\sigma_1' \to \cdots \to \sigma_n' \to \phi'\}, \phi' \rangle.$$

Proof: Immediately by Definition 10.8. ■

10.4 Expansion

The operation of expansion on types defined here corresponds to the notion of expansion as defined for '$\vdash_{\mathrm{BCD}}$' in [55] and for '$\vdash_{\mathrm{S}}$' in [6]. A difference between the notions of expansion as defined in [21] and [55] is that in those papers a *set of types involved in the expansion* is created. Here just type-variables are collected which gives a less complicated definition of expansion.

Definition 10.11 *i*) If Γ is a context and $\sigma \in \mathcal{T}$, then $\mathcal{T}_{\mathrm{s}}\langle \Gamma,\sigma \rangle$ is the set of all strict subtypes occurring in the pair $\langle \Gamma,\sigma \rangle$.

ii) The *last type-variable* of a strict type σ, *last-tv* (σ), is defined by:

$$\begin{array}{rcl} \textit{last-tv}\,(\varphi) & = & \varphi \\ \textit{last-tv}\,(\sigma \to \phi) & = & \textit{last-tv}\,(\phi) \end{array}$$

An expansion is essentially an operation on derivations characterised by the quadruple $\langle \phi, n, \Gamma, \sigma \rangle$, where ϕ triggers the expansion, n is the number of copies required, and $\langle \Gamma,\sigma \rangle$ is the context and type of the conclusion of the derivation involved.

Definition 10.12 For every ϕ, $n \geq 2$, context Γ, and σ, the quadruple $\langle \phi, n, \Gamma, \sigma \rangle$ determines an *expansion* $\mathit{Exp}_{\langle \phi,n,\Gamma,\sigma \rangle} : \mathcal{T} \to \mathcal{T}$, that is constructed as follows.

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

   a) If φ occurs in ϕ, then $\varphi \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$.

   b) If $\psi \in \mathcal{T}_\mathrm{s}\langle\Gamma,\sigma\rangle$, and $\textit{last-tv}\,(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, then for all type-variables φ that occur in ψ: $\varphi \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$.

ii) Suppose $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle = \{\varphi_1,\ldots,\varphi_m\}$. Choose $m \times n$ different type-variables $\varphi_1^1, \ldots, \varphi_n^1, \ldots, \varphi_1^m, \ldots, \varphi_n^m$, such that each φ_i^j does not occur in $\langle\Gamma,\sigma\rangle$, for $i \in \underline{n}$ and $j \in \underline{m}$. Let, for $i \in \underline{n}$, S_i be the substitution such that $S_i\,(\varphi_j) = \varphi_i^j$, for $j \in \underline{m}$.

iii) $Exp_{\langle\phi,n,\Gamma,\sigma\rangle}\,(\tau)$ is obtained by traversing τ top-down and replacing every subtype ϕ by $S_1\,(\phi)\cap\cdots\cap S_n(\phi)$, if $\textit{last-tv}\,(\phi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, i.e. (where $Ex = Exp_{\langle\phi,n,\Gamma,\sigma\rangle}$):

$$\begin{array}{lll}
Ex\,(\tau_1\cap\cdots\cap\tau_n) & = \cap_n(Ex\,(\psi_i)) & \\
Ex\,(\psi) & = \cap_n(S_i\,(\psi)) & \textit{if last-tv}\,(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle \\
Ex\,(\rho\to\psi) & = Ex\,(\rho) \to Ex\,(\psi) & \textit{if last-tv}\,(\psi) \notin \mathcal{V}_\phi\langle\Gamma,\sigma\rangle \\
Ex\,(\varphi) & = \varphi & \textit{if } \varphi \notin \mathcal{V}_\phi\langle\Gamma,\sigma\rangle
\end{array}$$

iv) $Ex\,(\Gamma') = \{x{:}Ex\,(\rho) \mid x{:}\rho \in \Gamma'\}$.

v) $Ex\,(\langle\Gamma',\sigma'\rangle) = \langle Ex\,(\Gamma'),Ex\,(\sigma')\rangle$.

Instead of $Exp_{\langle\phi,n,\Gamma,\sigma\rangle}$, we will write $\langle\phi,n,\Gamma,\sigma\rangle$.

Example 10.13 Let ϕ be the type $(\varphi_1\to\varphi_2)\to(\varphi_3\to\varphi_1)\to\varphi_3\to\varphi_2$, and Ex be the expansion $\langle\varphi_1,2,\emptyset,\phi\rangle$. Then $\mathcal{V}_{\varphi_1}\langle\emptyset,\phi\rangle = \{\varphi_1,\varphi_3\}$, and

$$Ex\,(\phi) = ((\varphi_4\cap\varphi_5)\to\varphi_2)\to(\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5)\to\varphi_6\cap\varphi_7\to\varphi_2.$$

For an operation of expansion the following property holds:

Lemma 10.14 Let $Ex = \langle\phi,n,\Gamma,\sigma\rangle$ *be an expansion, then* $Ex\,(\tau) = \cap_n\tau_i$ *with for every* $i \in \underline{n}$, τ_i *is a trivial variant of* τ, *or* $Ex\,(\tau) \in \mathcal{T}_\mathrm{s}$.

Proof: Immediately by Definition 10.12. ■

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 10.15 Let $\Gamma' \vdash_\mathrm{R} A{:}\psi$, *and* $\langle\phi,n,\Gamma,\sigma\rangle$ *be such that* $\mathcal{T}_\mathrm{s}\langle\Gamma',\psi\rangle \subseteq \mathcal{T}_\mathrm{s}\langle\Gamma,\sigma\rangle$. *If* $\textit{last-tv}\,(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, *then* $\varphi' \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$ *for every type-variable* φ' *that occurs in* $\langle\Gamma',\psi\rangle$.

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

i) $A \equiv x$, then $\Gamma' = \{x{:}\psi\}$. Since $\textit{last-tv}\,(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, and $\psi \in \mathcal{T}_\mathrm{s}\langle\{x{:}\psi\},\psi\rangle \subseteq \mathcal{T}_\mathrm{s}\langle\Gamma,\sigma\rangle$, all type-variables that occur in ψ are in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$.

ii) $A \equiv \lambda x.A'$, then $\psi = \rho\to\phi$, and $\Gamma',x{:}\rho \vdash_\mathrm{R} A'{:}\phi$ (if $\rho = \omega$, then $\Gamma',x{:}\rho = \Gamma'$). Since $\textit{last-tv}\,(\rho\to\phi) = \textit{last-tv}\,(\phi)$, and also $\mathcal{T}_\mathrm{s}\langle\Gamma' \cup \{x{:}\rho\},\phi\rangle \subseteq \mathcal{T}_\mathrm{s}\langle\Gamma',\rho\to\phi\rangle \subseteq \mathcal{T}_\mathrm{s}\langle\Gamma,\sigma\rangle$, we get, by induction, that all type-variables in $\langle\Gamma' \cup \{x{:}\rho\},\phi\rangle$ are in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$. So all type-variables in $\langle\Gamma',\rho\to\phi\rangle$ are in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$.

iii) $A \equiv xA_1\cdots A_m$. Then there are τ_j,Γ_j $(j \in \underline{m})$, such that $\Gamma_j \vdash_\mathrm{R} A_j{:}\tau_j$ for every $j \in \underline{m}$, and $\Gamma' = \cap_m\Gamma_i \cap \{x{:}\tau_1\to\cdots\to\tau_m\to\psi\}$. Since $\textit{last-tv}\,(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, and

$$\tau_1\to\cdots\to\tau_m\to\psi \in \mathcal{T}_\mathrm{s}\langle\cap_m\Gamma_i \cap \{x{:}\tau_1\to\cdots\to\tau_m\to\psi\},\psi\rangle \subseteq \mathcal{T}_\mathrm{s}\langle\Gamma,\sigma\rangle,$$

every type-variable in $\tau_1\to\cdots\to\tau_m\to\psi$ is in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$. If, for $j \in \underline{m}$, $\tau_j = \cap_{i\in k_j}\tau_j^i$, then, for every $l \in \underline{k_j}$, $\textit{last-tv}\,(\tau_j^l) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, and

$$\mathcal{T}_s\langle\Gamma_j,\tau_j^l\rangle \subseteq \mathcal{T}_s\langle\cap_m\Gamma_i\cap\{x{:}\tau_1\to\cdots\to\tau_m\to\psi\},\psi\rangle \subseteq \mathcal{T}_s\langle\Gamma,\sigma\rangle,$$

so all type-variables in $\langle\Gamma_j,\tau_j^l\rangle$ are in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, for $j\in\underline{m}$, $l\in\underline{k_j}$. So all type-variables in $\langle\cap_m\Gamma_i\cap\{x{:}\tau_1\to\cdots\to\tau_m\to\psi\},\psi\rangle$ are in $\mathcal{V}_\phi\langle\Gamma,\sigma\rangle$. ■

The next lemma shows that an expansion is either complete, i.e. affects all type-variables, or maps a strict type in a pair to a strict type.

Lemma 10.16 *Let $\Gamma'\vdash_R A{:}\psi$, and $Ex=\langle\phi,n,\Gamma,\sigma\rangle$ be such that $\mathcal{T}_s\langle\Gamma',\psi\rangle\subseteq\mathcal{T}_s\langle\Gamma,\sigma\rangle$. Then either there are ψ_i,Γ_i ($i\in\underline{n}$), such that $Ex(\langle\Gamma',\psi\rangle)=\langle\cap_n\Gamma_i,\tau_1\cap\cdots\cap\tau_n\rangle$ and, for every $i\in\underline{n}$, $\langle\Gamma_i,\psi_i\rangle$ is a trivial variant of $\langle\Gamma',\psi\rangle$, or $Ex(\langle\Gamma',\psi\rangle)=\langle\Gamma'',\psi'\rangle$, with $\psi'\in\mathcal{T}_s$.*

Proof: By Lemmas 10.14, and 10.15. ■

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

The following property is needed for the proofs that expansion is sound (Theorem 10.18), and for completeness of chains of operations (Theorem 10.27).

Property 10.17 *Let Ex be an expansion, ϕ such that last-tv(ϕ) is not affected by Ex, and $Ex(\phi)=\phi'$. Then:*

i) *$Ex(\langle\Gamma\cup\{x{:}\tau\},\phi\rangle)=\langle\Gamma'\cup\{x{:}\tau'\},\phi'\rangle$, iff $Ex(\langle\Gamma,\tau\to\phi\rangle)=\langle\Gamma',\tau'\to\phi'\rangle$.*

ii) *Let $Ex(\langle\Gamma_i,\sigma_i\rangle)=\langle\Gamma_i',\sigma_i'\rangle$, for every $i\in\underline{n}$. Then*

$$Ex(\langle\cap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\phi\},\phi\rangle) = \langle\cap_n\Gamma_i'\cap\{x{:}\sigma_1'\to\cdots\to\sigma_n'\to\phi'\},\phi'\rangle.$$

Proof: Easy. ■

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

Theorem 10.18 *If $\Gamma\vdash_R A{:}\sigma$ and let Ex be such that $Ex(\langle\Gamma,\sigma\rangle)=\langle\Gamma',\sigma'\rangle$, then $\Gamma\vdash_R A{:}\sigma$.*

Proof: Without loss if generality, assume $\sigma=\psi\in\mathcal{T}_s$. By Lemma 10.16, we have two cases:

i) $\sigma'=\cap_m\psi_i'$, $\Gamma'=\cap_m\Gamma_i$, and each $\langle\Gamma_i,\psi_i\rangle$ is a trivial variant of $\langle\Gamma,\psi\rangle$ and, therefore, $\Gamma_i\vdash_R A{:}\psi_i$. So, by rule $(\cap I)$, $\Gamma\vdash_R A{:}\sigma$.

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

$(A\equiv\lambda x.A')$: Then $\psi=\rho\to\phi$, and $\Gamma,x{:}\rho\vdash_R A'{:}\phi$. Let $\psi'=\rho'\to\phi'$. (Notice that, if $\rho=\omega$, by Definition 10.12, also $\rho'=\omega$). By Lemma 10.17(*i*), $Ex(\langle\Gamma\cup\{x{:}\rho\},\phi\rangle)=\langle\Gamma'\cup\{x{:}\rho'\},\phi'\rangle$, and, by induction, $\Gamma',x{:}\rho'\vdash_R A'{:}\phi'$, so $\Gamma'\vdash_R \lambda x.A'{:}\rho'\to\phi'$.

$(A\equiv xA_1\cdots A_n$, *with* $n\geq 0)$: Then $\Gamma=\cap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\phi\}$, and, for $i\in\underline{n}$, $\Gamma_i\vdash_R A_i{:}\sigma_i$. Let, for $i\in\underline{n}$, $Ex(\langle\Gamma_i,\sigma_i\rangle)=\langle\Gamma_i',\sigma_i'\rangle$, then $\Gamma_i'\vdash_R A_j{:}\sigma_i'$ by induction. Then, by Lemma 10.17(*ii*),

$$Ex(\langle\cap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\phi\},\phi\rangle) = \langle\cap_n\Gamma_i'\cap\{x{:}\sigma_1'\to\cdots\to\sigma_n'\to\phi'\},\phi'\rangle.$$

Then $\cap_n\Gamma_i'\cap\{x{:}\sigma_1'\to\cdots\to\sigma_n'\to\phi'\}\vdash_R xA_1\cdots A_m{:}\phi'$. ■

10.5 Covering

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 '$\lessdot$' defined on pairs. This relation is inspired by the relation '$\sqsubseteq$' on terms in $\mathcal{A}$, and the relation between the principal pairs of two terms that are in that relation (see also Theorem 10.28).

Definition 10.19 The relation on pairs '$\lessdot$' is defined by:

$$\begin{array}{rcll}
\langle \Gamma,\sigma\rangle & \prec\!\!\prec & \langle \emptyset,\omega\rangle & \\
\langle \cap_n\Gamma_i,\phi_1\cap\cdots\cap\phi_n\rangle & \prec\!\!\prec & \langle \cap_n\Gamma'_i,\sigma'_1\cap\cdots\cap\sigma'_n\rangle & \textit{iff}\ \forall i\in \underline{n}\ (n\geq 2)\ [\,\langle\Gamma_i,\phi_i\rangle \prec\!\!\prec \langle\Gamma'_i,\phi'_i\rangle\,] \\
\langle \Gamma,\rho\to\psi\rangle & \prec\!\!\prec & \langle \Gamma',\rho'\to\psi'\rangle & \textit{iff}\ \langle\Gamma\cup\{x{:}\rho\},\psi\rangle \prec\!\!\prec \langle\Gamma'\cup\{x{:}\rho'\},\psi'\rangle \\
\langle \cap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\sigma\},\sigma\rangle & \prec\!\!\prec & \langle \cap_n\Gamma'_i\cap\{x{:}\sigma'_1\to\cdots\to\sigma'_n\to\sigma\},\sigma\rangle & \\
& & & \textit{iff}\ \forall i\in \underline{n}\ [\,\langle\Gamma_i,\sigma_i\rangle \prec\!\!\prec \langle\Gamma'_i,\sigma'_i\rangle\,]
\end{array}$$

Definition 10.20 A *covering Cov* is an operation denoted by a pair of pairs $\prec\langle\Gamma_0,\tau_0\rangle,\langle\Gamma_1,\tau_1\rangle\succ$ such that $\langle\Gamma_0,\tau_0\rangle \prec\!\!\prec \langle\Gamma_1,\tau_1\rangle$, and is defined by:

$$\begin{array}{rcl}
\mathit{Cov}\,(\langle\Gamma,\sigma\rangle) & = & \langle\Gamma_1,\tau_1\rangle,\ \textit{if}\ \langle\Gamma,\sigma\rangle = \langle\Gamma_0,\tau_0\rangle, \\
& = & \langle\Gamma,\sigma\rangle,\ \ \textit{otherwise}.
\end{array}$$

The operation of covering is not sound as a general operation for '$\vdash_{\mathrm{E}}$'.

Example 10.21 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 for x since $x{:}\alpha\cap(\alpha\to\alpha) \vdash_{\mathrm{E}} x{:}\alpha$ is derivable, but we cannot derive $x{:}\omega\to\alpha \vdash_{\mathrm{E}} x{:}\alpha$.

The operation of covering is, however, sound for '$\vdash_{\mathrm{R}}$'.

Theorem 10.22 *For every covering Cov, if* $\mathit{Cov}\,(\langle\Gamma,\sigma\rangle) = \langle\Gamma',\sigma'\rangle$, *and* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$, *then* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$.

Proof: By induction on the structure of types.

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

ii) $\sigma' = \sigma'_1\cap\cdots\cap\sigma'_n$, $n\geq 2$. Then $\Gamma = \cap_n\Gamma_i$, $\sigma = \phi_1\cap\cdots\cap\phi_n$, and, for every $i\in\underline{n}$, $\Gamma_i \vdash_{\mathrm{R}} A{:}\phi_i$. By Definition 10.19, $\Gamma' = \cap_n\Gamma'_i$, and, for every $i\in\underline{n}$, $\prec\langle\Gamma_i,\phi_i\rangle,\langle\Gamma'_i,\phi'_i\rangle\succ$ is a covering. Then, by induction, $\Gamma'_i \vdash_{\mathrm{R}} A{:}\phi'_i$, so also $\Gamma' \vdash_{\mathrm{R}} A{:}\sigma'_1\cap\cdots\cap\sigma'_n$.

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

$(A \equiv \lambda x.A')$: If $\Gamma \vdash_{\mathrm{R}} \lambda x.A'{:}\sigma$, $\sigma = \rho\to\phi$, $\Gamma,x{:}\rho \vdash_{\mathrm{R}} A'{:}\phi$, and $\sigma' = \rho'\to\phi'$. Since we know that $\prec\langle\Gamma,\rho\to\phi\rangle,\langle\Gamma',\rho'\to\phi'\rangle\succ$ is a covering, also $\prec\langle\Gamma\cup\{x{:}\rho\},\phi\rangle,\langle\Gamma'\cup\{x{:}\rho'\},\phi'\rangle\succ$ is a covering, so, by induction, $\Gamma',x{:}\rho' \vdash_{\mathrm{R}} A'{:}\phi'$, so $\Gamma' \vdash_{\mathrm{R}} \lambda x.A'{:}\rho'\to\phi'$.

$(A \equiv yA_1\cdots A_n$, *with* $n\geq 0)$: Since $\Gamma \vdash_{\mathrm{R}} yA_1\cdots A_n{:}\sigma$, there are σ_i,Γ_i $(i\in\underline{n})$, such that $\Gamma = \cap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\sigma\}$, and $\Gamma_i \vdash_{\mathrm{R}} A_i{:}\sigma_i$, for every $i\in\underline{n}$. By Definition 10.19, there are σ'_i,Γ'_i $(i\in\underline{n})$, such that $\prec\langle\Gamma_i,\sigma_i\rangle,\langle\Gamma'_i,\sigma'_i\rangle\succ$ is a covering for every $i\in\underline{n}$, and $\Gamma' = \cap_n\Gamma'_i\cap\{x{:}\sigma'_1\to\cdots\to\sigma'_n\to\sigma'\}$. Then by induction, for $i\in\underline{n}$, $\Gamma'_i \vdash_{\mathrm{R}} A_i{:}\sigma'_i$, so also, by $(\to E)$, $\cap_n\Gamma'_i\cap\{x{:}\sigma'_1\to\cdots\to\sigma'_n\to\phi'\} \vdash_{\mathrm{R}} yA_1\cdots A_n{:}\sigma'$. ■

10.6 Lifting

The last operation needed in this section is that of lifting:

Definition 10.23 A *lifting L* is denoted by a pair of pairs $\langle\langle\Gamma_0,\tau_0\rangle,\langle\Gamma_1,\tau_1\rangle\rangle$ such that $\tau_0 \leq_{\mathrm{E}} \tau_1$ and $\Gamma_1 \leq_{\mathrm{E}} \Gamma_0$, and is defined by:

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

ii) $L\,(\Gamma) = \Gamma_1$, if $\Gamma = \Gamma_0$; otherwise, $L\,(\Gamma) = \Gamma$.

iii) $L\,(\langle\Gamma,\sigma\rangle) = \langle L\,(\Gamma),L\,(\sigma)\rangle$.

The operation of lifting is clearly not sound for '$\vdash_{\mathrm{R}}$', since we can derive $\vdash_{\mathrm{R}} \lambda x.x{:}\phi\to\phi$, and $\phi\to\phi \leq_{\mathrm{E}} \phi\cap\psi\to\phi$, but we cannot derive $\vdash_{\mathrm{R}} \lambda x.x{:}\phi\cap\psi\to\phi$.

The operation of lifting is sound for '$\vdash_{\mathrm{E}}$'.

Theorem 10.24 *If* $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$ *and* $\langle\langle\Gamma,\sigma\rangle,\langle\Gamma',\sigma'\rangle\rangle$ *is a lifting, then* $\Gamma' \vdash_{\mathrm{E}} M{:}\sigma'$.

Proof: By Definition 10.23, $\Gamma' \leq_{\rm E} \Gamma$, and $\sigma \leq_{\rm E} \sigma'$. The proof follows from Lemma 4.4. ■

10.7 Completeness of operations

We will now show completeness of the above specified operations, both for '$\vdash_{\rm R}$' as for '$\vdash_{\rm E}$'. First the notion of chain of operations is introduced.

Definition 10.25 *i*) A *chain of operations* is an object $[O_1,\ldots,O_n]$, where each O_i is an expansion, covering, substitution, or lifting, and

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

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: $\overrightarrow{Ex_i} * [S,Cov]$.

iv) An *essential chain* is a relevant chain, right-concatenated with at most one 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 10.10, and 10.17 are needed for the inductive step in case of an abstraction term, part (*ii*) of those lemmas are needed for the inductive step in case of an application term. Notice that, by construction, all operations mentioned in that part satisfy the conditions required by these lemmas.

Theorem 10.26 (Completeness) *If $\Gamma \vdash_{\rm R} A{:}\sigma$ and $pp\,(A) = \langle\Pi,\pi\rangle$, then there exists a relevant chain Ch such that $Ch\,(\langle\Pi,\pi\rangle) = \langle\Gamma,\sigma\rangle$.*

Proof: By induction on the definition of types

$(\sigma = \omega)$: Take $Cov = \prec\langle\Pi,\pi\rangle,\langle\emptyset,\omega\rangle\succ$, which, by Definition 10.20, is a covering. Take $Ch = [Cov]$.

$(\sigma = \phi_1\cap\cdots\cap\phi_n)$: Then $\Gamma = \cap_n\Gamma_i$, for some Γ_i $(i \in \underline{n})$, and $\Gamma_i \vdash_{\rm R} A{:}\phi_i$, for $i \in \underline{n}$. Let $Ex = \langle\pi,n,\Pi,\pi\rangle$, then $Ex\,(\langle\Pi,\pi\rangle) = \langle\cap_n\Pi_i,\cap_n\pi_i\rangle$, with $pp\,(A) = \langle\Pi_i,\pi_i\rangle$. By induction, there exist relevant chains $Ch_1,\ldots,Ch_n$ such that for $i \in \underline{n}$, $Ch_i\,(\langle\Pi_i,\pi_i\rangle) = \langle\Gamma_i,\phi_i\rangle$. Let, for $i \in \underline{n}$, $Ch_i = \overrightarrow{Ex_i} * [S_i] * [Cov_i]$, and Γ_i',ϕ_i' be such that $\overrightarrow{Ex_i} * [S_i]\,(\langle\Pi_i,\pi_i\rangle) = \langle\Gamma_i',\phi_i'\rangle$, and $Cov_i = \prec\langle\Gamma_i',\phi_i'\rangle,\langle\Gamma_i,\phi_i\rangle\succ$. Then, by Definition 10.19,

$$Cov = \prec\langle\cap_n\Gamma_i',\tau_1'\cap\cdots\cap\tau_n'\rangle,\langle\cap_n\Gamma_i,\cap_n\phi_i\rangle\succ$$

is a covering. Take $Ch = [Ex] * \overrightarrow{Ex_1} * \cdots * \overrightarrow{Ex_n} * [S_1 \circ \cdots \circ S_n] * [Cov]$.

$(\sigma \in \mathcal{T}_{\rm s})$: This part is proven by induction on the structure of elements of $\mathcal{A}$. Notice that the case that $A \equiv \bot$ need not be considered.

$(A \equiv x)$: Then $\Gamma = \{x{:}\sigma\}$, $\Pi = \{x{:}\varphi\}$, and $\pi = \varphi$. Take $Ch = [(\varphi \mapsto \sigma)]$.

$(A \equiv \lambda x.A')$: *1*) $x \in FV(A')$. Then $\sigma = \alpha\to\phi$, $\Gamma,x{:}\alpha \vdash_{\rm R} A'{:}\phi$, and $pp\,(\lambda x.A') = \langle\Pi,\mu\to\pi\rangle$, where $pp\,(A') = \langle\Pi\cup\{x{:}\mu\},\pi\rangle$. By induction there exists a relevant chain $Ch' = \overrightarrow{Ex} * [S] * [Cov']$ such that $Ch'\,(\langle\Pi\cup\{x{:}\mu\},\pi\rangle) = \langle\Gamma\cup\{x{:}\alpha\},\phi\rangle$. Let α',ϕ',Γ' be such that $\overrightarrow{Ex} * [S]\,(\langle\Pi\cup\{x{:}\mu\},\pi\rangle) = \langle\Gamma'\cup\{x{:}\alpha'\},\phi'\rangle$, and

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

Since $\phi \in \mathcal{T}_{\rm s}$, by construction also $\phi' \in \mathcal{T}_{\rm s}$ and, by Lemmas 10.17(*i*) and 10.10(*i*), $\overrightarrow{Ex} * [S]\,(\langle\Pi,\mu\to\pi\rangle) = \langle\Gamma',\alpha'\to\phi'\rangle$. Take $Cov = \prec\langle\Gamma',\alpha'\to\phi'\rangle,\langle\Gamma,\alpha\to\phi\rangle\succ$, which, by Definition 10.19, is a covering. Take $Ch = \overrightarrow{Ex} * [S] * [Cov]$.

2) $x \notin FV(A')$. Then there $\sigma = \omega{\rightarrow}\phi$, $\Gamma \vdash_{\mathrm{R}} A' : \phi$, and $pp\,(\lambda x.A') = \langle \Pi, \omega{\rightarrow}\pi \rangle$, where $pp\,(A') = \langle \Pi, \pi \rangle$. By induction there exists a relevant chain $Ch' = \overrightarrow{Ex} * [S] * [Cov']$ such that $Ch'\,(\langle \Pi, \pi \rangle) = \langle \Gamma, \phi \rangle$. Let Γ', ϕ' be such that $\overrightarrow{Ex} * [S]\,(\langle \Pi, \pi \rangle) = \langle \Gamma', \phi' \rangle$, and $Cov' = \prec\langle \Gamma', \phi' \rangle, \langle \Gamma, \phi \rangle\succ$. Since $\phi \in \mathcal{T}_{\mathrm{s}}$, by construction also $\phi' \in \mathcal{T}_{\mathrm{s}}$ and, by Lemma 10.17(*i*), and 10.10(*i*), $\overrightarrow{Ex} * [S]\,(\langle \Pi, \omega{\rightarrow}\pi \rangle) = \langle \Gamma', \omega{\rightarrow}\phi' \rangle$. Take

$$Cov = \prec\langle \Gamma', \omega{\rightarrow}\phi' \rangle, \langle \Gamma, \omega{\rightarrow}\phi \rangle\succ,$$

which, by Definition 10.19, is a covering. Take $Ch = \overrightarrow{Ex} * [S] * [Cov]$.

$(A \equiv xA_1 \cdots A_m)$: Then there are σ_i, Γ_i $(i \in \underline{m})$, such that $\Gamma_j \vdash_{\mathrm{R}} A_j : \sigma_j$, for every $j \in \underline{m}$, $\Gamma = \cap_m \Gamma_j \cap \{x{:}\sigma_1{\rightarrow}\cdots{\rightarrow}\sigma_m{\rightarrow}\sigma\}$, and $\Pi = \cap_m \Pi_j \cap \{x{:}\pi_1{\rightarrow}\cdots{\rightarrow}\pi_m{\rightarrow}\varphi\}$, $\pi = \varphi$, with for every $j \in \underline{m}$, $pp\,(A_j) = \langle \Pi_j, \pi_j \rangle$, in which φ does not occur. By induction, there are relevant chains $Ch_1, \ldots, Ch_m$ such that, for $j \in \underline{m}$, $Ch_j\,(\langle \Pi_j, \pi_j \rangle) = \langle \Gamma_j, \sigma_j \rangle$. Let, for $j \in \underline{m}$, $Ch\,(j) = \overrightarrow{Ex}_j * [S_j] * [Cov_j]$, and Γ'_j, σ'_j be such that $\overrightarrow{Ex}_j * [S_j]\,(\langle \Pi_j, \pi_j \rangle) = \langle \Gamma'_j, \sigma'_j \rangle$, and $Cov_j = \prec\langle \Gamma'_j, \sigma'_j \rangle, \langle \Gamma_j, \sigma_j \rangle\succ$. Let $\overrightarrow{Ex} = \overrightarrow{Ex}_1 * \cdots * \overrightarrow{Ex}_m$, and $S = S_1 \circ \cdots \circ S_m \circ (\varphi \mapsto \sigma)$, then, because of Lemmas 10.17(*ii*) and 10.10(*ii*):

$$\begin{aligned}\overrightarrow{Ex} * [S]\,(\langle \cap_m \Pi_j \cap \{x{:}\pi_1{\rightarrow}\cdots{\rightarrow}\pi_m{\rightarrow}\varphi\}, \varphi \rangle) \\ = \langle \cap_m \Gamma'_j \cap \{x{:}\sigma'_1{\rightarrow}\cdots{\rightarrow}\sigma'_m{\rightarrow}\sigma\}, \sigma \rangle.\end{aligned}$$

Then, by Definition 10.19,

$$Cov = \prec\langle \cap_m \Gamma'_j \cap \{x{:}\sigma'_1{\rightarrow}\cdots{\rightarrow}\sigma'_m{\rightarrow}\sigma\}, \sigma \rangle, \langle \cap_m \Gamma_i \cap \{x{:}\phi_1{\rightarrow}\cdots{\rightarrow}\phi_m{\rightarrow}\sigma\}, \sigma \rangle\succ,$$

is a covering. Take $Ch = \overrightarrow{Ex} * [S] * [Cov]$. ■

Notice that, in the proof above, we needed to separate the empty intersection. Should we treat ω as the empty intersection, then, following the structure of the proof, ω would be (implicitly) dealt with by the second case $(\sigma = \phi_1 \cap \cdots \cap \phi_n)$; then the chains that follow from induction do not exist, so no chain is produced, and the principal type of A is not changed, and the type ω will not be created.

We can now show a completeness result for '$\vdash_{\mathrm{E}}$':

Theorem 10.27 *If $\Gamma \vdash_{\mathrm{E}} A : \sigma$ and $pp\,(A) = \langle \Pi, \pi \rangle$, then there exists an essential chain Ch such that $Ch\,(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.*

Proof: By Lemma 10.3 there are Γ', σ' such that $\Gamma' \vdash_{\mathrm{R}} A : \sigma$, $\sigma' \leq_{\mathrm{E}} \sigma$, and $\Gamma \leq_{\mathrm{E}} \Gamma'$. By Theorem 10.26, there exists a relevant chain Ch such that such that $Ch\,(\langle \Pi, \pi \rangle) = \langle \Gamma', \sigma' \rangle$. Since $\langle \langle \Gamma', \sigma' \rangle, \langle \Gamma, \sigma \rangle \rangle$ is a lifting, by Definition 10.25(*iv*), there exists an essential chain such that $Ch\,(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$. ■

Like in [22, 55, 6], we can prove that there exists a precise relation between terms in $\mathcal{A}$ and principal pairs, both equipped with an appropriate ordering. This relation is in [55] 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 10.19 is used.

Theorem 10.28 *$\langle \mathcal{P}, \succ\!\!\succ \rangle$ is a meet semi-lattice isomorphic to $\langle \mathcal{A}, \sqsubseteq \rangle$.*

Proof: pp is, as function from $\mathcal{A}$ to $\mathcal{P}$, by Definition 10.6(*ii*), surjective. It is injective because of Theorems 7.4, 7.6, and 10.27. That pp respects the order, i.e. if $A \sqsubseteq A'$, then $pp\,(A') \prec\!\!\prec pp\,(A)$, follows by straightforward induction. ■

Definition 10.29 *i*) Let M be a term. Let $\mathcal{P}(M)$ be the set of all principal pairs for all approximants of M: $\mathcal{P}(M) = \{pp\,(A) \mid A \in \mathcal{A}(M)\}$.

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

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

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

Since by Lemma 6.1, type assignment is closed for '$\sqsubseteq$', for $A \sqsubseteq A'$, the principal type for A can be generated (by covering) from the principal pair of A'. So, for a term with a finite set of approximants, the principal pair for the most 'detailed' approximant (the largest) suffices, but for a term with infinite approximants, this will not work.

The proof of the principal pair property for '$\vdash_{\mathrm{E}}$' is completed by the following:

Theorem 10.30 *Let Γ and σ be such that $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$.*

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

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

Proof: From $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$ and Theorem 6.9 follows that there $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\mathrm{E}} A{:}\sigma$. Then, by Definition 10.29, either

i) there exists $A_M \in \mathcal{A}(M)$ such that $pp\,(M) = pp\,(A_M) = \langle \Pi,\pi \rangle$. Since $pp\,(A_M)$ is minimal in $\mathcal{P}$, so $pp\,(A_M) \prec\!\!\prec pp\,(A)$, by Theorem 10.28, A_M is maximal in $\mathcal{A}(M)$, so $A \sqsubseteq A_M$. Then, by Lemma 6.1, also $\Gamma \vdash_{\mathrm{E}} A_M{:}\sigma$.

ii) or $\langle \Pi,\pi \rangle = pp\,(A) \in \mathcal{P}(M)$.

In any case, by Theorem 10.27, there exists an essential chain such that $Ch\,(\langle \Pi,\pi \rangle) = \langle \Gamma,\sigma \rangle$. ■

The same result, using relevant chains rather than essential chains, can be formulated for '$\vdash_{\mathrm{R}}$'.

This last result shows that terms with a finite set of approximants (a finite Böhm-tree) have a single principal pair, and that terms with an infinite set of approximants (a infinite Böhm-tree) have a infinitely many principal pairs, one for each approximant.

11 Concluding remarks

Intersection type assignment system comes in many shapes and forms. Originally set up by Coppo and Dezani in '78 to characterise normalisation, the concept quickly evolved into the well-know system defined by Barendregt, Coppo and Dezani of '83. This was shown to have many strong properties, including giving rise to a filter semantics and completeness, and characterising head-normalisation and normalisation. Other important elementary properties, like the approximation result and the principal pairs property were later shown by Ronchi della Rocca and Venneri in '84.

A few years later, in '88 I myself discovered not only the strength of intersection types, but also their beauty, especially when proving the strong normalisation result for the BCD-system. However, I also started to question the superfluousness of certain points, like BCD's approach of treating intersection as a generic type constructor, or ω as a type constant, and subsequently defined the subset of strict intersection types, in order to come to a more syntax directed system without losing any of the main properties.

The strict intersection system, the first system I defined – which turned out to be very close to a system already considered before by Coppo, Dezani, and Venneri in '80 and '81– proved to be as expressive as the BCD-system, in that in '88 I showed that it types the same terms (with equivalent types). However, it differs in that it is not closed for η-reduction, and the corresponding strict filter model, which is essentially Engeler's model, does not express extensionality. I showed that this strict system has the principal pair property in '91, where the lack of extensionality of the system created complications in the proofs.

In '92 I noticed that, by adding contra-variance to the type-inclusion relation, extensionality could be achieved without resorting to the full BCD-system: this lead to the definition of the essential intersection system, for which I managed to show all major characterisation and termination results, as well as the principal pair property. In '95, in order to show the characterisation of head-normalisation, and normalisation, I first showed the approximation result.

After having proven a number of characterisation and termination result using Tait's technique, I could not but observe that these proofs had much structure in common, and looked for a common factor; this turned out to be the strong normalisation of derivation reduction. In '01 I showed this for the strict system, and showed that all characterisation results follow from this main result; in '05 I found the solution for the essential system in the definition of the $\leq$-relation on derivations as defined here.

I have put all these results in order and in context in this survey, thus not only putting the strength of strict types (within the essential system) into evidence, but also their elegance.

References

[1] F. Alessi, F. Barbanera, and M. Dezani-Ciancaglini. Intersection Types and Computational Rules. *Electronic Notes in Theoretical Computer Science*, 84, 2003.

[2] S. van Bakel. Derivations in Type Assignment Systems. Master's thesis, University of Nijmegen, 1988.

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

[4] S. van Bakel. Essential Intersection Type Assignment. In R.K. Shyamasunda, editor, *Proceedings of FST&TCS'93. 13th 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.

[5] S. van Bakel. *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*. PhD thesis, Department of Computer Science, University of Nijmegen, 1993.

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

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

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

[9] S. van Bakel. Rank 2 Types for Term Graph Rewriting (Extended Abstract). In *Electronic Proceedings of International Workshop* Types in Programming *(TIP'02), Dagstuhl, Germany*, volume 75 of *Electronic Notes in Theoretical Computer Science*, 2002.

[10] S. van Bakel. Cut-Elimination in the Strict Intersection Type Assignment System is Strongly Normalising. *Notre Dame journal of Formal Logic*, 45(1):35–63, 2004.

[11] S. van Bakel. The Heart of Intersection Type Assignment; Normalisation proofs revisited. *Theoretical Computer Science*, 398:82–94, 2008.

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

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

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

[15] 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.

[16] P. N. Benton. *Strictness Analysis of Lazy Functional Programs*. PhD thesis, Computing Lab, University of Cambridge, 1993. Technical Report 309.

[17] Luca Cardelli. Basic polymorphic typechecking. *Science of Computer Programming*, 8(2):147–172, 1987.

[18] A. Church. A note on the entscheidungsproblem. *Journal of Symbolic Logic*, 1(1):40–41, 1936.

[19] M. Coppo, M. Dezani, and P. Sallé. Functional characterization of some semantic Equalities inside λ-calculus. In H.A. Maurer, editor, *Proceedings of ICALP'79. 6th International Colloquium on Automata, Languages and Programming,* Graz, Austria, number 71 in Lecture Notes in Computer Science, pages 133–146. Springer-Verlag, 1979.

[20] 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.

[21] 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.

[22] 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.

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

[24] M. Coppo and A. Ferrari. Type inference, abstract interpretation and strictness analysis. In M. Dezani-Ciancaglini, S. Ronchi Della Rocca, and M. Venturini Zilli, editors, *A Collection of contributions on honour of Corrado Böhm*, pages 113–145. Elsevier, 1993.

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

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

[27] L. Damas and R. Milner. Principal type-schemes for functional programs. In *Proceedings 9th ACM Symposium on Principles of Programming Languages*, pages 207–212, 1982.

[28] F. Damiani. Typing Local Definitions and Conditional Expressions with Rank 2 Intersection. In *Proceedings of FOSSACS'00*, volume 1784 of *Lecture Notes in Computer Science*, pages 82–97. Springer-Verlag, 2000.

[29] 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.

[30] Ferruccio Damiani. Rank 2 intersection types for local definitions and conditional expressions. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 25(4):401–451, 2003.

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

[32] M. Dezani-Ciancaglini, R.K. Meyer, and Y. Motohama. The semantics of entailment omega. *Notre Dame journal of Formal Logic*, 43(3):129–145, 2002.

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

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

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

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

[37] 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.

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

[39] J.R. Hindley. *Basic Simple Type Theory*. Cambridge University Press, 1997.

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

[41] P. Hudak, S. Peyton Jones, P. Wadler, B. Boutel, J. Fairbairn, J. Fasel, K. Hammond, J. Hughes, T. Johnsson, D. Kieburtz, R. Nikhil, W. Partain, and J. Peterson. Report on the Programming Language Haskell. *ACM SIGPLAN Notices*, 27(5):1–64, 1992.

[42] T. Jensen. *Abstract Interpretation in Logical Form*. PhD thesis, Imperial College, University of London, November 1992.

[43] Thomas P. Jensen. Conjunctive type systems and abstract interpretation of higher-order functional programs. *Journal of Logic and Computation*, 5(4):397–421, 1995.

[44] T. Jim. What Are Principal Typings and What Are They Good For? In *Proceedings of the 23rd ACM symposium on Principles of Programming Languages (POPL '96)*, pages 42–53, 1996.

[45] A. Kfoury and J. Wells. Principality and decidable type inference for finite-rank intersection types. In *Proceedings of the 26th ACM Symposium on the Principles of Programming Languages (POPL '99)*, pages 161–174, 1999.

[46] A.J. Kfoury, H.G. Mairson, F.A. Turbak, and J.B. Wells. Relating Typability and Expressibility in Finite-Rank Intersection Type Systems. In *Proceedings of ICFP'99, International Conference on Functional Programming*, pages 90–101, 1999.

[47] D. Leivant. Polymorphic Type Inference. In *Proceedings 10th ACM Symposium on Principles of Programming Languages*, pages 88–98, Austin Texas, 1983.

[48] R. Milner. A Theory of Type Polymorphism in Programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.

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

[50] C. Retoré. A note on intersection types. INRIA Rapport de recherche 2431, INRIA, France, 1994.

[51] 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.

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

[53] J. A. Robinson. A Machine-Oriented Logic Based on Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[54] S. Ronchi Della Rocca. Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59:181–209, 1988.

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

[56] 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. Springer-Verlag, 1978.

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

[58] Tachio Terauchi and Alex Aiken. On typability for rank-2 intersection types with polymorphic recursion. In *LICS*, pages 111–122, 2006.

[59] 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.

[60] J.B. Wells. The essence of principal typings. In *Proceedings of ICALP'92. 29th International Colloquium on Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 913–925. Springer-Verlag, 2002.