Strict Intersection Types for the Lambda Calculus

STEFFEN VAN BAKEL, Imperial College London

This article will show the usefulness and elegance of strict intersection types for the Lambda Calculus, that are strict in the sense that they are the representatives of equivalence classes of types in the BCD-system [Barendregt et al. 1983]. We will focus on the essential intersection type assignment; this system is almost syntax directed, and we will show that all major properties hold that are known to hold for other intersection systems, like the approximation theorem, the characterization of (head/strong) normalization, completeness of type assignment using filter semantics, strong normalization for cut-elimination and the principal pair property. In part, the proofs for these properties are new; we will briefly compare the essential system with other existing systems.

Categories and Subject Descriptors: F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages

General Terms: Theory

Additional Key Words and Phrases: Strict intersection types, λ-calculus, filter semantics, principal pairs, approximation, normalization

ACM Reference Format:
Bakel, S. van. 2011. Strict intersection types for the Lambda Calculus. ACM Comput. Surv. 43, 3, Article 20 (April 2011), 49 pages.
DOI = 10.1145/1922649.1922657 http://doi.acm.org/10.1145/1922649.1922657

1. INTRODUCTION

In 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 [Curry and Feys 1958] (see also Hindley [1997]). Formulated via natural deduction,[1] using the arrow type constructor '$\rightarrow$', it expresses abstraction and application, and allows for a basic functional characterization of terms. It is well known that in Curry's system, the problem of typeability:

$$\text{Given a term } M, \text{ are there a context } \Gamma \text{ and a type } \sigma \text{ such that } \Gamma \vdash M : \sigma?$$

is decidable; for this reason it gained popularity as the basis for modern type systems for functional programming languages, like Haskell [Hudak et al. 1992] and ML [Milner 1978]. Although Curry's system is already powerful, in that many programs can be

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

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

DOI 10.1145/1922649.1922657 http://doi.acm.org/10.1145/1922649.1922657

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 cannot be assigned the same types (see Section 3.1).

The Intersection Type Discipline is an extension of Curry's system that does not have these drawbacks. The extension consists mainly of allowing for term-variables (and terms) to have more than one type, joined via the intersection type constructor '$\cap$', as well as adding 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 characterization 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 Coppo and Dezani-Ciancaglini [1980], which introduced intersection types[3] ('$\vdash_{\text{CD}}$', see Definition 3.2). From this initial system several others emerged; the best known and most frequently quoted is the BCD-system ('$\vdash_{\text{BCD}}$', see Definition 3.10) as presented by Barendregt, Coppo, and Dezani-Ciancaglini in Barendregt et al. [1983], but there are two earlier papers [Coppo et al. 1981, 1980] that investigate systems that can be regarded as in-between '$\vdash_{\text{CD}}$' and '$\vdash_{\text{BCD}}$'. In the first of these papers, 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 contain the type constant 'ω'.[4] The first system presented in Coppo et al. [1981] ('$\vdash_{\text{CDV}}$', 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 generalizes 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 '$\leq$', which, is contravariant in arrow types, and type assignment is closed for this relation.

As mentioned, the slight generalization 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 characterized 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 Barendregt et al. [1983] 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 (see Property 3.15).

Another way to define semantics for the λ-calculus is through approximation: as in Wadsworth [1976], and Barendregt [1984], 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 [Barendregt 1984]. It is well known that interpreting γ-terms by their Böhm tree gives a model for the Lambda Calculus, and the same is possible when interpreting a term M through its set of approximants, $\mathcal{A}(M)$. From the Approximation Theorem, the observation that there exists a very precise

[2]This problem is normally addressed via more expressive type systems, using polymorphism and recursion [Damas and Milner 1982], 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."

[4]The type constant ω was first introduced by Sallé [1978]; a joint paper appeared as Coppo et al. [1979].

relation between types assignable to a term and those assignable to its approximants, formulated as:

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

(see Coppo et al. [1980], Ronchi Della Rocca and Venneri [1984], and van Bakel [1992, 1995] and Sections 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 Barendregt et al. [1983]; van Bakel [1992, 1995] and Section 7.1).

Another reason for the popularity of Curry's system within the context of programming languages is that it enjoys 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 Hindley [1969] in the context of Combinator Systems, successfully exploiting for the first time, the notion of Robinson [1965] concerning unification in the context of type theory. The principal type property found its way into programming, mainly through the pioneering work of Milner [1978], (see also Damas and Milner [1982]). 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, that is, functions that can be applied to various kinds of arguments, even of incomparable type. The formal motivation of this concept lies directly in the notion of principal types [Cardelli 1987].

A disadvantage of '$\vdash_{\text{BCD}}$' (and of any real intersection system, for that matter) is that type assignment in that system 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 [van Bakel 1993b], as first suggested by Leivant [1983], which 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 $(\sigma_1 \cap \cdots \cap \sigma_n) \to \tau$, where the σ_i are types that do not contain intersections. This kind of type was later used outside the context of the λ-calculus [van Bakel 1996, 2002; Damiani 2000, 2003; Terauchi and Aiken 2006]; many decidable restrictions of various ranks were later defined by Kfoury and Wells [1999], and Kfoury et al. [1999].

That intersection types can be used as a basis for programming languages, was first discussed by Reynolds [1981]. This led to the development of the typed (i.e. terms contain types syntactically) programming language, Forsythe [Reynolds 1988], 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 also found their use in the context of abstract interpretation [Jensen 1992, 1995; Coppo and Ferrari 1993; Benton 1993].

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 Barendregt et al. [1983], the relation $\leq$ induces an equivalence relation $\sim$ on types; equivalence classes are big (for example: $\omega \sim \sigma \to \omega$, for all types σ) and type assignment is closed for $\sim$. This makes the problem of inhabitation (is there a closed term that has this type) complex.

[5]In the past, the terminology principal type, was used often but principal pair expresses this property better; see also Wells [2002] and Jim [1996] for discussions on principal types, pairs, and typings.

[6]This type system is sometimes also referred to as the Hindley-Milner system.

Building on the definition of principal context and type scheme introduced in Coppo et al. [1980], Ronchi Della Rocca and Venneri [1984] 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 is complicated:

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

This is not only caused by the undecidability of the problem—even a semialgorithm 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 van Bakel [1992] (a first version appeared in van Bakel [1988], which coined the moniker *strict*) is a restriction of $\vdash_{\text{BCD}}$, which uses strict types, a variant of intersection types that has been used in many papers since, which are actually the normalized tail-proper types of Coppo et al. [1981]. In spite of the 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 normalizable terms and those having a normal form can be equally elegantly characterized. 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, which satisfies all major properties of the filter λ-model as presented in Barendregt et al. [1983], but is an essentially different λ-model, equivalent to the model $\mathcal{D}_A$ of Engeler [1981]. With the use of the inference type semantics in van Bakel [1992], soundness and completeness for strict type assignment was proven, without having to introduce $\leq$ (see also Section 5).

This article 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 van Bakel [1993b, 1995] (albeit in different notation), and later studied in van Bakel [2008]; we will revisit the results of those papers, give some in part new proofs, and briefly compare it with other existing systems. $\vdash_{\text{E}}$ is a true restriction of $\vdash_{\text{BCD}}$, which satisfies all properties of that system, and is an extension of $\vdash_{\text{CD}}$. It is also an extension of $\vdash_{\text{S}}$; the major difference is that it, like $\vdash_{\text{BCD}}$, will prove to be closed for η-reduction: if $\Gamma \vdash_{\text{E}} M{:}\sigma$ and $M \to_\eta N$, then $\Gamma \vdash_{\text{E}} N{:}\sigma$.

By establishing all major properties in the context of strict types, we will in fact show that the treatment of intersection types as in Barendregt et al. [1983] has been too general; the same results can be obtained for a far less complicated system, that more closely follows 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 in van Bakel [1995] (see Section 10) and 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 Barendregt et al. [1983]; this result can also be obtained for $\vdash_{\text{E}}$ (see Section 7.2).

Other main results for intersection systems, like the Approximation Theorem and Strong Normalization Theorem were proven for $\vdash_{\text{E}}$ independently van Bakel [1995, 1992], though both used the technique of Computability Predicates of Tait [1967]. This technique has been widely used to study normalization properties or similar results, as for example in Coppo et al. [1987], Dezani-Ciancaglini and Margaria [1986], and Ronchi Della Rocca [1988]. In this survey, we will show that both these properties are special cases of a more fundamental result, being the strong normalization result for derivation reduction (cut-elimination). The proof for this property uses a variant

of the technique developed in van Bakel and Fernández [1997] for Term Rewriting, that has also found its application in the field of Combinator Systems in [van Bakel and Fernández 2003]. This more fundamental result was first published in van Bakel [2008]; a similar result for $\vdash_{\mathrm{S}}$ appeared as van Bakel [2004].

For intersection systems, there is a significant difference between derivation reduction and ordinary reduction (see Section 9.2); unlike normal typed or type assignment systems, in $\vdash_{\mathrm{E}}$ not every term redex occurs with types in a derivation. Moreover, especially the use of a contravariant relation $\leq$ on types, together with an associated derivation rule, greatly disturbs the smoothness of proofs (see again Section 9.2).

From this strong normalization result for derivation reduction, the Approximation Theorem and Strong Normalization Theorem, follow easily. The first of these implies the Head-Normalization Theorem and indirectly, the Normalization Theorem, as was already shown in van Bakel [1995] (see Section 6).

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

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

These properties and their proofs will be reviewed here.

1.1. Article Outline

In Section 2, we repeat some definitions for the λ-calculus that are relevant to this article, and introduce the notion of approximation and approximants. In Section 4, we will recall the definition of the essential type assignment system of van Bakel [1995], 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, 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 characterization of head-normalizability and normalizability. The relation between the essential and the BCD-systems 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 normalizable 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 normalization result in Section 9.3. This result will lead in Section 9.4 to new proofs for the approximation and strong-normalization results; we will briefly discuss the proof for the strong normalization of derivation reduction in the strict and in the relevant systems. 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 previously appeared in van Bakel [1988, 1992, 1993b, 1993c, 1993a, 1995, 2004, 2008].

Because of space restrictions, all proofs appear in the electronic appendix available in the ACM Digital Library.

1.2. 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, and Q are used for λ-terms; x, y, and z for term-variables; $M[N/x]$ for the usual operation of substitution on λ-terms; and A for terms in $\lambda\perp$-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 $\vdash$, normally indexed when 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 $\vec{\ }$ for the purpose of abbreviation. For example, $P\vec{M_i}$ stands for $PM_1 \cdots M_n$ for a suitable n, and $[N_1/x_1, \ldots, N_n/x_n]$ is abbreviated by $[\vec{N_i/x_i}]$.

2. THE λ-CALCULUS

We assume the reader to be familiar with the λ-calculus [Church 1936; Barendregt 1984]; 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*).

(1) 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.

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

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

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

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

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

Definition 2.2 (β-*contraction*).

(1) The reduction relation $\rightarrow_\beta$ is defined as the contextual (i.e. compatible [Barendregt 1984]) closure of the rule:

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

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

(3) 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.

(1) The set $\mathcal{N} \subset \Lambda$ of *normal forms* is defined by the grammar:
$$N ::= xN_1 \cdots N_n \, (n \geq 0) \mid \lambda x.N.$$
(2) 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:

(1) *normalizable* if there exists an $N \in \mathcal{N}$ such that $M \twoheadrightarrow_\beta N$.
(2) *head-normalizable* if there exists an $H \in \mathcal{H}$ such that $M \twoheadrightarrow_\beta H$.
(3) *strongly normalizable* 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 Wadsworth [1976], and is defined using the notion of terms in $\lambda\bot$-normal form.[7]

Definition 2.5.

(1) The set $\Lambda\bot$ of $\lambda\bot$ *-terms* is defined by:
$$M ::= x \mid \bot \mid \lambda x.M \mid M_1M_2.$$
(2) The notion of reduction $\to_{\beta\bot}$ is defined as $\to_\beta$, extended by:
$$\begin{aligned} \lambda x.\bot &\to_{\beta\bot} \bot \\ \bot M &\to_{\beta\bot} \bot. \end{aligned}$$
(3) The set of *normal forms for elements of* $\Lambda\bot$, *with respect to* $\to_{\beta\bot}$, is the set $\mathcal{A}$ of $\lambda\bot$*-normal forms* or *approximate normal forms*, ranged over by A, defined by:
$$A ::= \bot \mid \lambda x.A \, (A \neq \bot) \mid xA_1 \cdots A_n \, (n \geq 0).$$

Definition 2.6 (*Approximants*).

(1) The partial order $\sqsubseteq \subseteq (\Lambda\bot)^2$ is defined as the smallest preorder (i.e. reflexive and transitive relation) such that:
$$\begin{aligned} \bot &\sqsubseteq M \\ M \sqsubseteq M' &\Rightarrow \lambda x.M \sqsubseteq \lambda x.M' \\ M_1 \sqsubseteq M_1' \ \& \ M_2 \sqsubseteq M_2' &\Rightarrow M_1M_2 \sqsubseteq M_1'M_2'. \end{aligned}$$
(2) For $A \in \mathcal{A}$, $M \in \Lambda$, if $A \sqsubseteq M$, then A is called a *direct approximant* of M.
(3) The relation $\precsim \subseteq \mathcal{A} \times \Lambda\bot$ is defined by:
$$A \precsim M \Leftrightarrow \exists M' \, [M' =_\beta M \ \& \ A \sqsubseteq M'].$$
If $A \precsim M$, then A is called an *approximant* of M.
(4) $\mathcal{A}(M) = \{A \in \mathcal{A} \mid A \precsim M\}$.

The following properties of approximants hold and are needed.

PROPOSITION 2.7.

(*1*) *If* $A \in \mathcal{A}(x\overrightarrow{M_i})$, $A \neq \bot$ *and* $A' \in \mathcal{A}(N)$, *then also* $AA' \in \mathcal{A}(x\overrightarrow{M_i}N)$.

[7] As in Barendregt [1984], we use $\bot$ (called *bottom*) instead of Ω; also, the symbol $\precsim$ is used for a relation on $\lambda\bot$-terms, inspired by a similar relation defined on Böhm-trees in Barendregt [1984].

(2) *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*
—$\lambda z.A \in \mathcal{A}(M)$.
(3) *If* $M =_\beta N$*, then* $\mathcal{A}(M) = \mathcal{A}(N)$.

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

Definition 2.8.

(1) On $\Lambda\bot$, the partial mapping *join*, $\sqcup : \Lambda\bot \times \Lambda\bot \to \Lambda\bot$, is defined by:

$$\begin{array}{rcl@{\qquad}rcl} \bot \sqcup M \equiv M \sqcup \bot & \equiv & M & (\lambda x.M) \sqcup (\lambda x.N) & \equiv & \lambda x.(M \sqcup N) \\ x \sqcup x & \equiv & x & (M_1M_2) \sqcup (N_1N_2) & \equiv & (M_1 \sqcup N_1)(M_2 \sqcup N_2) \end{array}$$

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

Notice that, for example, $(\lambda x.x) \sqcup (\lambda yz.yz)$ is undefined.

The last alternative in the definition of $\sqcup$ defines the join on applications in a more general way than that of Gunter and Scott [1990], which would state that:

$$(M_1M_2) \sqcup (N_1N_2) \sqsubseteq (M_1 \sqcup N_1)(M_2 \sqcup N_2),$$

since a join of two arbitrary terms does not always exist. 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(3), $\mathcal{A}(M)$ can be used to define a semantics for the Lambda Calculus. In fact, it is possible to show that:

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

where $\mathrm{BT}(M)$ stands for the *Böhm tree* of M, a tree that represents the (possible infinite) normal form of M (see Barendregt [1984]).

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:*

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

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 $M \equiv \bot$.

PROPOSITION 2.11.

(1) *If* $M \sqsubseteq M_i$*, for all* $i \in \underline{n}$*, then* $M \sqsubseteq M_1 \sqcup \cdots \sqcup M_n$.
(2) *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* $M_3 \sqsubseteq M_1$, $M_4 \sqsubseteq M_2$.

3. A HISTORICAL PERSPECTIVE

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

3.1. Curry Type Assignment

Type assignment for the Lambda Calculus was first studied by Curry [1934] (see also Curry and Feys [1958], Hindley [1997]). Curry's original system expresses abstraction and application, and types are built over the single type constant o using the type constructor $\to$ (arrow). The system we will present in this section is a generalization of that system in that types are built over type variables and terms have infinitely many types.

Definition 3.1.

(1) 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.

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

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

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

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

(3) 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.

(4) We write $\Gamma \vdash_{\mathrm{C}} M{:}\tau$ if there exists a derivation built using the preceding rules with conclusion $M : \tau$ and the context Γ contains at least all of the uncancelled statements in that derivation (notice that, in rule $(\to I)$, the statement $x{:}\sigma$ is cancelled).

As used in the preceding (and as also used in Coppo et al. [1981], Coppo and Dezani-Ciancaglini [1980], Barendregt et al. [1983], and van Bakel [1992, 1995]), the notation for the type assignment rules is that of natural deduction. A disadvantage of that notation is that the precise structure of contexts: 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 van Bakel [2004]), 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 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 following.

—*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, given a typeable term M, will return a pair $\langle \Gamma, \sigma \rangle$, such that $\Gamma \vdash_{\mathrm{C}} M{:}\sigma$ - in fact, it then returns the principal pair and will fail if M is untypeable.

—Strong normalizability 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, their principal type schemes 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 in the following is an extension of Curry's system for the pure Lambda Calculus, which does not have these drawbacks. It

has developed over a period of several years; in order to develop a concise overview of the field, various systems will be shown; their presentation is adjusted to our notation.

3.2. The Coppo-Dezani Type Assignment System

Intersection type assignment was first introduced by Coppo and Dezani-Ciancaglini [1980]. 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 $(\rightarrow I)$-derivation rule and therefore also allowing for more than one type for the right-hand term in the $(\rightarrow E)$-derivation rule.

Definition 3.2 (*[Coppo and Dezani-Ciancaglini 1980]*).

(1) The set of types is inductively defined by:

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

(2) *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 Coppo and Dezani-Ciancaglini [1980] are the following.

—*Subject reduction*. if $\Gamma \vdash_{\text{CD}} M{:}\sigma$, and $M \rightarrow_\beta N$, then $\Gamma \vdash_{\text{CD}} N{:}\sigma$.
—*Normalizability 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 Coppo et al. [1981], two type assignment systems are presented 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 Sallé [1978]. The first system presented in Coppo et al. [1981] is a true extension of the one presented in Definition 3.2. The second one is a relevant restriction.

Definition 3.3 (*[Coppo et al. 1981]*).

(1) The set of types is inductively defined by:

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

(2) Every nonintersection 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 $\phi \neq \omega$.
(3) *Type assignment* is defined by:

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

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

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

As remarked in the preceding, 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. In fact, the original presentation for rule $(\to I)$:

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

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 subterms 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.

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

Definition 3.4 (*Normalized types*). The set of *normalized types* is inductively defined in Coppo et al. [1981] by:

(1) All type-variables are normalized types.
(2) If σ is a normalized intersection type and ψ is a normalized type, then $\sigma \to \psi$ is a normalized type.
(3) ω is a normalized intersection type.
(4) If $\phi_1,\ldots,\phi_n$ are normalized types ($n \geq 1$), then $\phi_1 \cap \cdots \cap \phi_n$ is a normalized intersection type.

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

The main properties of this system proven in Coppo et al. [1981] are the following.

—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.

Normalized types play an important role in these characterization properties.

The second type assignment system presented in Coppo et al. [1981] is a restricted version of the first, by restricting it to normalized types and limiting the possible contexts that can be used in a derivation. This yields a system that 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: 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 by: $x:\cap_m \phi_j \in \bigcap\{\Gamma_1,\ldots,\Gamma_n\}$ if and only if $\{x:\phi_1,\ldots,x:\phi_m\}$ is the set of all statements on x that occur in $\Gamma_1 \cup \cdots \cup \Gamma_n$, and x occurs in this set, so $1 \leq m \leq n$.

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

Definition 3.6 (*[Coppo et al. 1981]*). *Restricted type assignment* is defined by:

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

$$(\to E):\frac{\Gamma_1\vdash M{:}\sigma\to\phi\quad\Gamma_2\vdash N{:}\sigma}{\bigcap\{\Gamma_1,\Gamma_2\}\vdash MN{:}\phi}\qquad(\cap I):\frac{\Gamma_1\vdash M{:}\phi_1\quad\cdots\quad\Gamma_n\vdash M{:}\phi_n}{\bigcap_n\Gamma_i\vdash M{:}\phi_1\cap\cdots\cap\phi_n}\,(n\geq 1).$$

We write $\Gamma\vdash_{\mathrm{R}} M{:}\sigma$ 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 preceding 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 $\vdash_{\mathrm{R}}$ proven in Coppo et al. [1981] are as follows.

—If $\Gamma\vdash_{\mathrm{R}} M{:}\sigma$, then σ is a normalized type.
—If $\Gamma\vdash_{\mathrm{R}} M{:}\sigma$ and $M\to_\eta N$, then $\Gamma\vdash_{\mathrm{R}} N{:}\sigma$.

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

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

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

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

and we can now derive

$$\dfrac{\dfrac{}{x{:}\sigma\to\phi\vdash x{:}\sigma\to\phi}}{\vdash\lambda x.x{:}(\sigma\to\phi)\to\sigma\to\phi}.$$

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

3.4. A System with Principal Pairs

Coppo et al. [1980] also present 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 [Robinson 1965] is the main tool to type an application [Hindley 1969], and type-substitution is the only operation used for the generation of pairs.

Instead, the principal type scheme for a term is in Coppo et al. [1980] (as well as in Ronchi Della Rocca and Venneri [1984], and van Bakel [1993c, 1995]; 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 an infinite number of approximants have infinitely many principal pairs. This can be understood from the observation that, using intersection types, 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):

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

($f^0\bot = \bot$, and $f^n\bot = f(f^{n-1}\bot)$). 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 $\Gamma = f{:}(\omega\to\phi_2)\cap(\phi_2\to\phi_1)$):

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

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 following operations. Therefore, a functional characterisation of these terms, through a principal pair, cannot be represented in a finite way.

The type assignment system studied in Coppo et al. [1980] is the restricted one from Coppo et al. [1981], but with the set of types of the unrestricted system. The reason to not use the normalized types as well, is the fact that ω is treated as a type constant; since ω can be substituted for φ in $\sigma\to\varphi$, $\sigma\to\omega$ 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 $(\rightarrow E)$-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. As an illustration, suppose the following is a correct derivation.

$$\dfrac{\dfrac{}{\Gamma, x{:}\sigma\rightarrow\phi \vdash x:\sigma\rightarrow\phi} \quad \boxed{}}{\Gamma, x{:}\sigma\rightarrow\phi \vdash xN:\phi}\,(\rightarrow E),$$

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

$$\dfrac{\dfrac{}{\Gamma', x{:}\,\cap_n\sigma_i\rightarrow\phi \vdash x:\,\cap_n\sigma_i\rightarrow\phi} \quad \dfrac{\boxed{} \;\cdots\; \boxed{}}{\Gamma' \vdash N:\,\cap_n\sigma_i}\,(\cap I)}{\Gamma', x{:}\,\cap_n\sigma_i\rightarrow\phi \vdash xN:\phi}\,(\rightarrow E).$$

An expansion indicates not only the type to be expanded, but also the number of copies that have 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 $\tau\rightarrow\mu$ is also a subtype of σ, it will be expanded into $\cap_n(\tau_i\rightarrow\mu_i)$, where the τ_i are copies of τ. Then τ is affected by the expansion of μ; all other occurrences 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 (see Definition 9.11).

The construction of the principal pair property follows the outline of Section 10; the technique used is the same as later used in Ronchi Della Rocca and Venneri [1984] and van Bakel [1993c]. It defines principal pairs of context and type for approximate normal forms, specifying the operations of expansion and substitution proved sufficient to generate all possible pairs for those terms from their principal pair, and generalizing this result to arbitrary λ-terms.

The set of ground pairs for a term $A\in\mathcal{A}$, as defined in Coppo et al. [1980], is proven to be complete for A, in the sense that all other pairs for A (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, 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 9.6), 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 the following.

—$pp(A)$ 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,

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

and (taking $\Gamma = f{:}(\omega\to\phi_2)\cap(\phi_2\to\phi_1)$),

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, x{:}\omega\to\phi_2 \vdash x:\omega\to\phi_2} \quad \dfrac{}{\Gamma, x{:}\omega\to\phi_2 \vdash x:\omega}(\omega)}{\Gamma, x{:}\omega\to\phi_2 \vdash xx:\phi_2}(\to E)}{\Gamma, x{:}\omega\to\phi_2 \vdash f:\phi_2\to\phi_1 \quad \vdots}(\to E)}{\Gamma, x{:}\omega\to\phi_2 \vdash f(xx):\phi_1}(\to I)}{\Gamma \vdash \lambda x.f(xx):(\omega\to\phi_2)\to\phi_1} \quad \dfrac{\dfrac{\dfrac{}{\Gamma, x{:}\omega \vdash f:\omega\to\phi_2} \quad \dfrac{}{\Gamma, x{:}\omega \vdash xx:\omega}(\omega)}{\Gamma, x{:}\omega \vdash f(xx):\phi_2}(\to E)}{\Gamma \vdash \lambda x.f(xx):\omega\to\phi_2}(\to I)$$

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

3.5. The Barendregt-Coppo-Dezani System

The type assignment system presented by Barendregt, Coppo and Dezani-Ciancaglini in [Barendregt et al. 1983] (the BCD-system) is based on $\vdash_{\text{CDV}}$. It strengthened that system by treating $\cap$ as a normal type constructor like $\to$, allowing intersections to occur at the right of arrow types, not just on the left. It also introduced a partial order relation $\leq$ on types, added the type assignment rule $(\leq)$ to close type assignment for this relation, and introduced a more general form of the rules concerning intersection. The set of types derivable for a λ-term in this extended system is a filter: 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 Barendregt et al. [1983] was to show completeness of type assignment, for which the type inclusion relation $\leq$ and the type assignment rule $(\leq)$ were added. This was achieved by showing the statement: $\Gamma \vdash_{\text{BCD}} M:\sigma \iff [\![M]\!]_{\xi_\Gamma} \in \upsilon(\sigma)$ (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; see Property 3.15), where $\upsilon : \mathcal{T}_{\text{BCD}} \to \mathcal{F}_{\text{BCD}}$ is a simple type interpretation as defined in Hindley [1983] (see Definition 3.14). In order to prove the $\Leftarrow$-part of this statement (completeness), the relation $\leq$ is needed.

[8]Called $\mathcal{F}$ in Barendregt et al. [1983], but subscripted here to be able to distinguish it from other filter models.

Definition 3.10.

(1) $\mathcal{T}_{\text{BCD}}$, the set of types in Barendregt et al. [1983] is defined by:
$$\sigma, \tau \ ::= \ \varphi \mid \omega \mid \sigma \to \tau \mid \sigma \cap \tau$$
(2) On $\mathcal{T}_{\text{BCD}}$, the type inclusion relation $\leq$ is defined as the smallest preorder such that:
$$\begin{array}{rcl} \sigma & \leq & \omega \\ \sigma\cap\tau \leq \sigma & & \sigma\cap\tau \leq \tau \\ \omega & \leq & \omega\to\omega \end{array} \qquad \begin{array}{rcl} (\sigma\to\tau)\cap(\sigma\to\rho) & \leq & \sigma\to\tau\cap\rho \\ \sigma \leq \tau \ \& \ \sigma \leq \rho & \Rightarrow & \sigma \leq \tau\cap\rho \\ \rho \leq \sigma \ \& \ \tau \leq \phi & \Rightarrow & \sigma\to\tau \leq \rho\to\phi \end{array}$$
(3) $\sim$ is the equivalence relation induced by '$\leq$': $\sigma \sim \tau \iff \sigma \leq \tau \leq \sigma$.
(4) 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 (*[Barendregt et al. 1983]*).
Type assignment is defined by the following sequent-style natural deduction system.
$$(\to I): \frac{\Gamma, x{:}\sigma \vdash M{:}\tau}{\Gamma \vdash \lambda x.M{:}\sigma\to\tau} \qquad (\to E): \frac{\Gamma \vdash M{:}\sigma\to\tau \quad \Gamma \vdash N{:}\sigma}{\Gamma \vdash MN{:}\tau}$$
$$(\cap I): \frac{\Gamma \vdash M{:}\sigma \quad \Gamma \vdash M{:}\tau}{\Gamma \vdash M{:}\sigma\cap\tau} \qquad (\cap E): \frac{\Gamma \vdash M{:}\sigma\cap\tau}{\Gamma \vdash M{:}\sigma} \quad \frac{\Gamma \vdash M{:}\sigma\cap\tau}{\Gamma \vdash M{:}\tau}$$
$$(\leq): \frac{\Gamma \vdash M{:}\sigma}{\Gamma \vdash M{:}\tau}\,(\sigma \leq \tau) \qquad (\omega): \frac{}{\Gamma \vdash M{:}\omega}$$
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 Barendregt et al. [1983] is a very easy type definition and easily understandable derivation rules. But this advantage is superficial, since all difficulties now show up while proving theorems; the complexity of $\leq$ and $\sim$ especially causes confusion.

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

Definition 3.12.

(1) 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) $\sigma \leq \tau \ \& \ \sigma \in d \Rightarrow \tau \in d$.
(2) $\mathcal{F}_{\text{BCD}} = \{d \mid d \text{ is a BCD-filter}\}$.
(3) 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 goal is to show that typeability implies satisfiability, and vice versa: $\Gamma \vdash M{:}\sigma \iff \Gamma \vDash M{:}\sigma$, where $\Gamma \vDash M{:}\sigma$ (semantic satisfiability) is a relation between the semantic interpretation of Γ, M, and σ, hence a notion of type interpretation is needed. As in Dezani-Ciancaglini and Margaria [1986] and Mitchell [1988], and essentially following Hindley [1982], a distinction can be made between,

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*).

(1) Let $\langle \mathcal{D}, \cdot, \varepsilon \rangle$ be a continuous λ-model. A mapping $\upsilon : \mathcal{T} \to \wp(\mathcal{D}) = \{X \mid X \subseteq \mathcal{D}\}$ is an *inference type interpretation* if and only if:
   (a) $\{\varepsilon \cdot d \mid \forall e \in \upsilon(\sigma)\,[d \cdot e \in \upsilon(\tau)]\} \subseteq \upsilon(\sigma \to \tau)$.
   (b) $\upsilon(\sigma \to \tau) \subseteq \{d \mid \forall e \in \upsilon(\sigma)\,[d \cdot e \in \upsilon(\tau)]\}$.
   (c) $\upsilon(\sigma \cap \tau) = \upsilon(\sigma) \cap \upsilon(\tau)$.
(2) Following Hindley [1983], a type interpretation is *simple* if also:
$$\upsilon(\sigma \to \tau) = \{d \mid \forall e \in \upsilon(\sigma)\,[d \cdot e \in \upsilon(\tau)]\}.$$
(3) A type interpretation is called an *F type interpretation* if it satisfies:
$$\upsilon(\sigma \to \tau) = \{\varepsilon \cdot d \mid \forall e \in \upsilon(\sigma)\,[d \cdot e \in \upsilon(\tau)]\}.$$

Notice that, in Part (2), the containment relation $\subseteq$ of Part (1).(b) is replaced by $=$, and that in Part (3) the same is done with regard to Part (1a).

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 υ a type interpretation. We define $\vDash$ by;

(1) $\mathcal{M}, \xi, \upsilon \vDash M : \sigma \iff [\![M]\!]^{\mathcal{M}}_{\xi} \in \upsilon(\sigma)$.
(2) $\mathcal{M}, \xi, \upsilon \vDash \Gamma \iff \mathcal{M}, \xi, \upsilon \vDash x{:}\sigma$ for every $x{:}\sigma \in \Gamma$.
(3) $\Gamma \vDash M{:}\sigma \iff \forall \mathcal{M}, \xi, \upsilon\ [\mathcal{M}, \xi, \upsilon \vDash \Gamma \Rightarrow \mathcal{M}, \xi, \upsilon \vDash M{:}\sigma]$
$\Gamma \vDash_{\mathrm{s}} M{:}\sigma \iff$
$\forall \mathcal{M}, \xi,$ *simple type interpretations* $\upsilon\,[\mathcal{M}, \xi, \upsilon \vDash \Gamma \Rightarrow \mathcal{M}, \xi, \upsilon \vDash M{:}\sigma]$.
$\Gamma \vDash_{\mathrm{F}} M{:}\sigma \iff \forall \mathcal{M}, \xi,$ *F type interpretations* $\upsilon\,[\mathcal{M}, \xi, \upsilon \vDash \Gamma \Rightarrow \mathcal{M}, \xi, \upsilon \vDash M{:}\sigma]$.

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

The following properties are proven in Barendregt et al. [1983].

—For all $M \in \Lambda$, $\{\sigma \mid \exists \Gamma\,[\Gamma \vdash_{\mathrm{BCD}} M{:}\sigma\,]\} \in \mathcal{F}_{\mathrm{BCD}}$.
—Let ξ be a valuation of term-variables in $\mathcal{F}_{\mathrm{BCD}}$ and $\Gamma_\xi = \{x{:}\sigma \mid \sigma \in \xi(x)\}$; for $M \in \Lambda$, define the interpretation of M in $\mathcal{F}_{\mathrm{BCD}}$ via $[\![M]\!]_\xi = \{\sigma \mid \Gamma_\xi \vdash_{\mathrm{BCD}} M{:}\sigma\}$. Using the method of Hindley and Longo [1980], it is shown that $\langle \mathcal{F}_{\mathrm{BCD}}, \cdot, [\![\cdot]\!] \rangle$ is a λ-model.

The main result of Barendregt et al. [1983] is obtained by proving the following.

Property 3.15.

(1) Soundness: $\Gamma \vdash_{\mathrm{BCD}} M{:}\sigma \Rightarrow \Gamma \vDash_{\mathrm{s}} M{:}\sigma$.
(2) 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 7.11. The results of Barendregt et al. [1983] show in fact that type assignment in $\vdash_{\mathrm{BCD}}$ is complete with respect to simple type semantics; this in contrast to $\vdash_{\mathrm{S}}$ (presented in van Bakel [1992], see also Section 5), which 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 not needed to solve the problem of completeness of type

assignment (see Property 3.15): the mere introduction of a relation on normalized types with contravariance in the arrow would have done the trick.

A characterization of strong normalisation for $\vdash_{\text{BCD}}$ is shown in van Bakel [1992], by proving that the set of all terms typeable by means of the derivation rules $(\cap I)$, $(\cap E)$, $(\to I)$ and $(\to E)$ is exactly the set of strongly normalizable terms. The proof for this property in that paper needs the rule $(\leq)$ for the contravariant $\leq$-relation. In van Bakel [2004], an alternative proof of strong normalization was given (much like that presented in Section 9, which appeared in van Bakel [2008]), but for the strict type assignment system of van Bakel [1992] (see Section 5); this proof does not need a contravariant $\leq$-relation.

3.6. Principal Pairs for $\vdash_{\text{BCD}}$

For $\vdash_{\text{BCD}}$, principal type schemes are defined by Ronchi Della Rocca and Venneri [1984]. 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, greatly disturbs the complexity of the definition of expansion, and through that, the accessibility of that paper.

The first operation defined, is substitution, which 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 generalization of the notion of expansion defined in Coppo et al. [1980, Section 3.4]; since intersections can now also appear on the right of an arrow, the construction of the set of types affected by an expansion is more involved. 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 9.6. To come to the proof of completeness of these operations Ronchi Della Rocca and Venneri [1984] first prove the approximation theorem for $\vdash_{\text{BCD}}$:

PROPERTY 3.16. *$\Gamma \vdash_{\text{BCD}} M{:}\sigma$ if and only if there exists $A \in \mathcal{A}(M)$ such that $\Gamma \vdash_{\text{BCD}} A{:}\sigma$.*

This is used in the proof of the principal type property, as in Theorem 9.25. The proof is completed by first proving that when $\mathcal{A}(M)$ is finite, then $\{pp(A) \mid A \in \mathcal{A}(M)\}$ has a maximal element (see Theorem 9.23 and Definition 9.24). The following is then proven.

PROPERTY 3.17.

(*1*) *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 $Ch(pp(A)) = \langle \Gamma, \sigma\rangle$.*

(*2*) *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 type assignment system as first presented in van Bakel [1995], a restricted version of $\vdash_{\text{BCD}}$, 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 $\vdash_{\text{S}}$ that was presented in van Bakel [1992] (see Section 5); the main difference is that $\vdash_{\text{S}}$ is not closed for η-reduction, whereas $\vdash_{\text{E}}$ is.

Strict types are the types that are strictly needed to assign a type to a term that is nontrivially typeable in $\vdash_{\text{BCD}}$. 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 a subtype at the left-hand side of an arrow-type scheme, as in the types of Coppo and Dezani-Ciancaglini[1980a] and Coppo et al. [1980, 1981], so ω does not occur in an intersection subtype or on the right of an arrow.[9]

Definition 4.1 (*Strict Types*).

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

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

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

(2) 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 $i \in \underline{n}$.

(3) The relation $\leq$ is defined as the least preorder on $\mathcal{T}$ such that:

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

(4) On $\mathcal{T}$, the relation $\sim_{\text{E}}$ is defined by: $\sigma \sim_{\text{E}} \tau \iff \sigma \leq_{\text{E}} \tau \leq_{\text{E}} \sigma$.

(5) The relations $\leq_{\text{E}}$ and $\sim_{\text{E}}$, are, as in Definition 3.10, extended to contexts.

Notice that, by the second clause of Part (3), $\sigma \leq_{\text{E}} \omega$, for all σ.

Compared to types defined previously, the set of types as considered in Coppo and Dezani-Ciancaglini [1980] are ω-free strict types (see Definition 8.1); which in Coppo et al. [1981] 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, normalizing 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 normalized types coincides with the set of strict types and normalized intersection types correspond to strict intersection types.

Since it is easy to show that intersection is commutative ($\sigma \cap \tau \sim_{\text{E}} \tau \cap \sigma$) and associative, ($\sigma \cap (\tau \cap \rho) \sim_{\text{E}} (\sigma \cap \tau) \cap \rho$), we can arbitrarily swap the order of subtypes in an intersection, and will write $\cap_n \phi_i$ for the type $\phi_1 \cap \cdots \cap \phi_n$, where $\cap_1 \phi_i = \phi_1$ and $\cap_0 \phi_i = \omega$.

[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 $\sigma \to \omega$ as a type.

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

$$\begin{array}{rcl}
\varphi \leq_{\mathrm{E}} \phi & \Longleftrightarrow & \phi \equiv \varphi. \\
\omega \leq_{\mathrm{E}} \sigma & \Longleftrightarrow & \sigma \equiv \omega. \\
\sigma \to \phi \leq_{\mathrm{E}} \rho \in \mathcal{T}_{\mathrm{S}} & \Longleftrightarrow & \rho \equiv \mu \to \psi \ \&\ \mu \leq_{\mathrm{E}} \sigma \ \&\ \phi \leq_{\mathrm{E}} \psi \\
\cap_n \phi_i \leq_{\mathrm{E}} \tau \in \mathcal{T}_{\mathrm{S}} & \Rightarrow & \exists i \in \underline{n}\, [\, \phi_i \leq_{\mathrm{E}} \tau] \\
\sigma \leq_{\mathrm{E}} \tau & \Rightarrow & \sigma = \cap_n \phi_i \ \&\ \tau = \cap_m \psi_j \ \&\ \forall j \in \underline{m}\ \exists i \in \underline{n}\, [\, \phi_i \leq_{\mathrm{E}} \psi_j\,]
\end{array}$$

Definition 4.3 (*Type assignment*).

(1) *Essential intersection type assignment* and *derivations* are defined by the following natural deduction system:

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

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

(2) We write $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$ if this statement is derivable using an essential intersection derivation, and $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M{:}\sigma$ 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 ϕ_i ($i \in \underline{n}$):

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

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 $\Gamma \vdash_{\mathrm{E}} M{:}\omega$, for all Γ, M, exactly as in $\vdash_{\mathrm{BCD}}$.

PROPOSITION 4.4. *For this notion of type assignment, the following properties hold:*

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

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

4.1. Eta Reduction

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

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

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

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* $\Gamma, x{:}\sigma \vdash_{\mathrm{E}} Mx{:}\phi$ *and* $x \notin fv(M)$ *then* $\Gamma \vdash_{\mathrm{E}} M{:}\sigma{\to}\phi$.

We can now show that $\vdash_{\mathrm{E}}$ is closed for η-reduction.

THEOREM 4.7 ($\vdash_{\mathrm{E}}$ CLOSED FOR $\to_\eta$). $\Gamma \vdash_{\mathrm{E}} M{:}\sigma \;\&\; M \to_\eta N \Rightarrow \Gamma \vdash_{\mathrm{E}} N{:}\sigma$.

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

Example 4.8. Notice that $\lambda xy.xy \to_\eta \lambda x.x$; then $\vdash_{\mathrm{E}} \lambda xy.xy{:}(\sigma{\to}\phi){\to}\sigma{\cap}\rho{\to}\phi$ and $\vdash_{\mathrm{E}} \lambda x.x{:}(\sigma{\to}\phi){\to}\sigma{\cap}\rho{\to}\phi$ are both easy to derive.

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

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

4.2. Subject Reduction and Expansion

As in Coppo et al. [1980] and Barendregt et al. [1983], it is possible to prove that $\vdash_{\mathrm{E}}$ 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. We will here prove this result in the same way (Corollary 7.10), 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 $(\to E)$, there exists τ such that

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

Since $(\to I)$ 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:

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

and these are all the applications of rule (Ax) that deal with x. Thus, for all $j{\in}\underline{m}$, $\tau \le_{\mathrm{E}} \psi_j$ 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 the basis of derivation reduction, as discussed in Section 9, and is formalized 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 $\phi_i\,(i \in \underline{n})$. A derivation for $\Gamma, x{:}\cap_n\phi_i \vdash_{\mathrm{E}} M:\psi$ can be obtained by replacing, in $\mathcal{D}$, all derivations for $\Gamma \vdash_{\mathrm{E}} N:\phi_i$ by the derivation for $x{:}\cap_n\phi_i \vdash_{\mathrm{E}} x:\phi_i$. Then, using $(\to I)$, $\Gamma \vdash_{\mathrm{E}} \lambda x.M:\cap_n\phi_i\to\psi$, and, using $(\cap I)$ on the collection of removed subderivations, $\Gamma\vdash_{\mathrm{E}} N:\cap_n\phi_i$. Then, using $(\to E)$, the redex can be typed.

When the term-variable x does not occur in M, the term N is a not a 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 $(\to I)$ gives $\Gamma \vdash_{\mathrm{E}} \lambda x.M:\omega\to\psi$. By $(\cap I)$, $\Gamma \vdash_{\mathrm{E}} N:\omega$, so, using $(\to E)$, 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, polymorphic type discipline Girard [1986], ML Damas and Milner [1982]), the implication holds in both directions.

LEMMA 4.9 (SUBSTITUTION LEMMA).

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

Since $\leq$ plays no role in the proof of this lemma, other than its $(\cap E)$ behaviour, 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 ($\vdash_{\mathrm{E}}$ CLOSED FOR $=_\beta$).
The following rules are admissible in $\vdash_{\mathrm{E}}$*:*

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

5. THE STRICT SYSTEM

Another system (in fact, the first) that uses strict intersection types is the strict system as presented in van Bakel [1992], in which the normalization properties and completeness are shown. For this system, the strong normalization of derivation reduction and characterization results can be proven using the same technique as in Section 9, as was first shown in van Bakel [2004]. It is defined as follows.

Definition 5.1. *Strict type assignment* is defined by the following natural deduction system:

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

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

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

We should emphasise the only difference between the essential type assignment of Definition 4.3 and the strict one: instead of the rule $(\cap E)$, it contains the rule (Ax), which uses a contravariant type inclusion relation. Notice that rule $(\cap E)$ is a special case of rule (Ax) in that $\cap_n\phi_i \leq_{\mathrm{E}} \phi_i$, for all $i \in \underline{n}$, so typeability in $\vdash_{\mathrm{S}}$ implies typeability in $\vdash_{\mathrm{E}}$.

van Bakel [1992] introduced a type inclusion relation $\leq_{\mathrm{S}}$ for this system, as the least preorder on $\mathcal{T}$ such that:

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

Notice that this relation expresses just the commutative and associative behavior of intersection, and excludes the contravariance. 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 the same system in terms of derivable judgements; rule $(\leq_{\mathrm{S}})$ combines $(\cap E)$ and $(\cap I)$. As for the difference in derivable statements between $\vdash_{\mathrm{S}}$ and $\vdash_{\mathrm{E}}$, it is possible to derive $\vdash_{\mathrm{E}} \lambda x.x : (\alpha \to \phi) \to (\alpha \cap \gamma) \to \phi$, which is not possible in $\vdash_{\mathrm{S}}$ (Example 4.8).

van Bakel [2004] 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 the model $\mathcal{D}_A$ Engeler [1981]. Using this model, in van Bakel [1992] 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 van Bakel [1992] shows that strict type assignment is sound and complete with respect to inference type semantics: $\Gamma \vdash_{\mathrm{S}} M : \sigma \iff \Gamma \vDash M : \sigma$ (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 van Bakel [1993c] is achieved in a way similar to, but significantly different from, the two techniques sketched in the preceding.

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 Ronchi Della Rocca and Venneri [1984], 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 Coppo et al. [1980], and Ronchi Della Rocca and Venneri [1984].

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 Coppo et al. [1980].) In that paper it 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 $\lambda\perp$-normal form, and $\langle \Pi, \pi \rangle$ is the principal pair for A, then there are a sequence of operations of expansion $\overrightarrow{E x_i}$, an operation of lifting L, and a substitution S, such that:

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

This result is then generalized to arbitrary λ-terms.

Because of technical reasons, substitution is defined in van Bakel [1993c] as the replacement of a type-variable φ by a type $\alpha \in \mathcal{T}_S \cup \{\omega\}$, 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 $\vdash_E$). 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 $\sigma \to \varphi$ (or $\sigma \cap \varphi$) into $\sigma \to \omega$ ($\sigma \cap \omega$), which is not a strict type. Therefore, for strict types, substitution is not defined as an operation that replaces type-variables by types, but as a mapping from types to types, that in a certain sense, normalizes while substituting.

The operation of expansion, as defined in van Bakel [1993c] (see Definition 9.11), corresponds to the one given in Coppo et al. [1980] and is a simplified version of the one defined in Ronchi Della Rocca and Venneri [1984]. A difference is that in those definitions subtypes are collected, whereas the definition of expansion in Definition 9.11 collects type-variables.

The operation of lifting as defined in van Bakel [1993c] (see Definition 9.18) is based on the relation $\leq_E$, 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.

$$\Gamma \vdash_S M{:}\sigma \ \& \ \sigma \leq_E \tau \ \Rightarrow \ \Gamma \vdash_S M{:}\sigma.$$

As a counterexample, take $x{:}\sigma \to \sigma \vdash_S x{:}\sigma \to \sigma$. Notice that $\sigma \to \sigma \leq_E \sigma \cap \tau \to \sigma$, but it is impossible to derive $x{:}\sigma \to \sigma \vdash_S x{:}\sigma \cap \tau \to \sigma$. (In fact, as argued in van Bakel [1993c], it is impossible to formulate an operation that performs the desired lifting and is sound on all pairs.) The reason for this is that applying a derivation rule that uses the relation $\leq_E$ corresponds to an η-reduction step (see Theorem 4.7), and $\vdash_S$ is not closed for η-reduction. Since strict type assignment is not closed for $\leq_E$, and the operation of lifting implicitly applies $\leq_E$ to a derivation, it is clear that a conflict arises.

However, in van Bakel [1993c], 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 Coppo et al. [1980]. 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 is then solved by allowing liftings only on primitive pairs for terms.

The result of van Bakel [1993c] follows from these results.

—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 $\Gamma \vdash_S A{:}\sigma$, then for all substitutions S: if $S(\langle \Gamma, \sigma \rangle) = \langle \Gamma', \sigma' \rangle$, then $\Gamma' \vdash_S A{:}\sigma'$.

Although lifting is not sound on all pairs, using these results, it is possible to prove that the three operations defined in van Bakel [1993c] are sufficient (complete). As in Ronchi Della Rocca and Venneri [1984], this result is then generalized to arbitrary λ-terms (see Property 3.17 and Theorem 9.21).

6. APPROXIMATION AND HEAD-NORMALIZATION RESULTS

In this section, we will prove a number of results for $\vdash_E$. First we will prove the approximation theorem (cf. Property 3.16); from this result, the well-known characterization

of head-normalization and normalization of λ-terms using intersection types follows easily. 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 Ronchi Della Rocca and Venneri [1984], the approximation result is obtained through a normalization of derivations, where all $(\rightarrow I)$–$(\rightarrow E)$ pairs, that derive a type for a redex $(\lambda x.M)N$, are replaced by one for its reduct $M[N/x]$, and all pairs of $(\cap I)$–$(\cap E)$ are eliminated. (This technique is also used in Coppo et al. [1980] and Barendregt et al. [1983]; it requires a rather complex notion of length of a derivation to show that this process terminates.) In this section, this result will be proven using the reducibility technique [Tait 1967]. With this result, we will show that $\vdash_{\mathrm{BCD}}$ is conservative over $\vdash_{\mathrm{E}}$, 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 $Appr(\Gamma, M, \sigma)$.

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

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

The following basic properties are needed further on.

LEMMA 6.2.

(1) $Appr(\Gamma, x\overrightarrow{M_i}, \sigma\rightarrow\phi) \;\&\; Appr(\Gamma, N, \sigma) \Rightarrow Appr(\Gamma, x\overrightarrow{M_i}N, \phi)$.
(2) $Appr(\Gamma \cup \{z{:}\sigma\}, Mz, \phi) \;\&\; z \notin fv(M) \Rightarrow Appr(\Gamma, M, \sigma\rightarrow\phi)$.
(3) $Appr(\Gamma, M[N/x]\vec{P}, \sigma) \Rightarrow Appr(\Gamma, (\lambda x.M)N\vec{P}, \sigma)$.

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 $Comp(\Gamma, M, \rho)$ is defined by induction over the structure of types:

$$\begin{aligned} Comp(\Gamma, M, \varphi) &\iff Appr(\Gamma, M, \varphi) \\ Comp(\Gamma, M, \sigma\rightarrow\phi) &\iff (\, Comp(\Gamma', N, \sigma) \Rightarrow Comp(\textstyle\bigcap\{\Gamma, \Gamma'\}, MN, \phi)\,) \\ Comp(\Gamma, M, \cap_n\phi_i) &\iff \forall i \in \underline{n}\,[\,Comp(\Gamma, M, \phi_i)\,] \end{aligned}$$

Notice that $Comp(\Gamma, M, \omega)$ holds (unconditionally) 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:

PROPOSITION 6.4.

(1) *If* $Comp(\Gamma, M, \sigma)$, *and* $\Gamma' \leq_{\mathrm{E}} \Gamma$, *then* $Comp(\Gamma', M, \sigma)$.
(2) *If* $Comp(\Gamma, M, \sigma)$, *and* $\sigma \leq_{\mathrm{E}} \tau$, *then* $Comp(\Gamma, M, \tau)$.

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

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

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

THEOREM 6.6.

(1) $Appr(\Gamma, x\overrightarrow{M_i}, \rho) \Rightarrow Comp(\Gamma, x\overrightarrow{M_i}, \rho)$.
(2) $Comp(\Gamma, M, \rho) \Rightarrow Appr(\Gamma, M, \rho)$.

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 $Comp(\cap_n \Gamma_i, M[\overrightarrow{N_i/x_i}], \sigma)$.*

This theorem then leads to the approximation result.

THEOREM 6.9 (APPROXIMATION THEOREM).

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

6.2. The Relation Between $\vdash_{\mathrm{BCD}}$ and $\vdash_{\mathrm{E}}$

We first show that $\vdash_{\mathrm{E}}$ is the nucleus of $\vdash_{\mathrm{BCD}}$ (see Section 3.5): we will show that for any derivation in $\vdash_{\mathrm{BCD}}$, it is possible to find an equivalent derivation in $\vdash_{\mathrm{E}}$. The proof is based on the fact that for every $\sigma \in \mathcal{T}_{\mathrm{BCD}}$, there is a $\sigma^* \in \mathcal{T}$ (the normalized version of σ) such that $\sigma \sim_{\mathrm{E}} \sigma^*$, and the Approximation Theorem 6.9.

Definition 6.10. *[Hindley 1982].*

(1) For every $\sigma \in \mathcal{T}_{\mathrm{BCD}}$, $\sigma^* \in \mathcal{T}$ is inductively defined as follows:

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

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

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

PROPOSITION 6.11.

(1) *[Hindley 1982] For every $\sigma \in \mathcal{T}_{\mathrm{BCD}}$, $\sigma \sim \sigma^*$.*
(2) *$\mathcal{T}_{\mathrm{BCD}}/\sim$ is isomorphic to $\mathcal{T}/\sim_{\mathrm{E}}$.*
(3) $\sigma \leq \tau \Rightarrow \sigma^* \leq_{\mathrm{E}} \tau^*$.
(4) $\sigma \in \mathcal{T} \Rightarrow \sigma = \sigma^*$.

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 $\vdash_{\mathrm{BCD}}$, a derivation in $\vdash_{\mathrm{E}}$ can be built, for which context and type in the conclusion are equivalent, and afterwards generalizing this result to arbitrary λ-terms. This depends on the approximation theorem for $\vdash_{\mathrm{BCD}}$ (Property 3.16) and Theorem 6.9.

We can directly link $\vdash_{\mathrm{BCD}}$ and $\vdash_{\mathrm{E}}$ on approximants via normalized types as follows.

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

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

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

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

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

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

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

6.3. Characterisation of Head-Normalization

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

THEOREM 6.15 (HEAD-NORMALIZATION). $\exists \Gamma, \phi\, [\Gamma \vdash_{\mathrm{E}} M{:}\phi] \iff$ *M has a head-normal form.*

To prepare the characterization of normalizability 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 normalizable terms are typeable without ω.

LEMMA 6.16.

(1) *If $\Gamma \vdash_{\mathrm{E}} A{:}\sigma$ and Γ, σ are ω-free, then A is $\bot$-free.*
(2) *If A is $\bot$-free, then there are ω-free Γ and σ, such that $\Gamma \vdash_{\mathrm{E}} A{:}\sigma$.*

By construction of the proof, in Part 6.16, 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 normalization result, Theorem 8.7.

Now, as shown in van Bakel [1992] for $\vdash_{\mathrm{S}}$ and in Barendregt et al. [1983] for $\vdash_{\mathrm{BCD}}$, it is possible to prove that we can characterize normalization.

THEOREM 6.17 (NORMALIZATION). $\exists \Gamma, \sigma\, [\Gamma \vdash_{\mathrm{E}} M{:}\sigma$ & Γ, σ *ω-free*$] \iff$ *M has a normal form.*

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 $\Gamma = \{x{:}(\alpha\to\beta\to\gamma)\cap\alpha,\, y{:}(\gamma\to\delta)\cap\beta\}$):

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

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

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

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:

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

Notice that this term is not strongly normalizable, 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 preceding proofs is to cover terms that will disappear during reduction. In view of these normalization results, 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, $\mathcal{D}::\vdash_{\mathrm{E}}\Theta\Theta:(\omega\to\phi)\to\phi$, 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.

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

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

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

(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 Barendregt et al. [1983] for $\vdash_{\mathrm{BCD}}$ (see Property 3.15), in this section we will show that $\vdash_{\mathrm{E}}$ is sound and complete with respect to the simple type semantics in Section 7.2; a similar result was shown in van Bakel [1992] for $\vdash_{\mathrm{S}}$, albeit with respect to inference type semantics, as we will discuss in Section 7.3.

7.1. The Essential Filter model

As in Barendregt et al. [1983] (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 $\leq_{\mathrm{E}}$.

Definition 7.1 (*Essential Filters*).

(1) A subset d of $\mathcal{T}$ is an *essential filter* if and only if:

$$\begin{array}{rcl} \phi_i \in d\ (\forall i \in \underline{n}, n \geq 0) & \Rightarrow & \cap_n \phi_i \in d \\ \tau \in d\ \&\ \tau \leq_{\mathrm{E}} \sigma & \Rightarrow & \sigma \in d \end{array}$$

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

(3) The domain $\mathcal{F}_{\mathrm{E}}$ is defined by: $\mathcal{F}_{\mathrm{E}} = \{d \subseteq \mathcal{T} \mid d \text{ is an essential filter}\}$. Application on $\mathcal{F}_{\mathrm{E}}$ is defined by:

$$d \cdot e = \uparrow_{\mathrm{E}}\{\phi \mid \exists \sigma \in e\ [\sigma \to \phi \in d]\}.$$

Notice that this definition is much the same as Definition 3.12, with the exception of the last part, where this definition forces the creation of a filter. Contrary to the case for $\vdash_{\mathrm{BCD}}$, application must be forced to yield filters, since in each arrow-type scheme $\sigma \to \phi \in \mathcal{T}$, ϕ is strict, and filters need to be closed for intersection.

$\langle \mathcal{F}_{\mathrm{E}}, \subseteq \rangle$ 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(1), and that an essential filter is a BCD filter, but not vice-versa.

For essential filters the following properties hold.

LEMMA 7.2.

(*1*) $\sigma \in \uparrow_{\mathrm{E}}\tau \iff \tau \leq_{\mathrm{E}} \sigma$.
(*2*) $\sigma \in \uparrow_{\mathrm{E}}\{\phi \mid \Gamma \vdash_{\mathrm{E}} M{:}\phi\} \iff \Gamma \vdash_{\mathrm{E}} M{:}\sigma$.

In particular, by part (2), $\{\sigma \mid \Gamma \vdash_{\mathrm{E}} M{:}\sigma\} \in \mathcal{F}_{\mathrm{E}}$.

Definition 7.3. The domain constructors $F : \mathcal{F}_{\mathrm{E}} \to [\mathcal{F}_{\mathrm{E}} \to \mathcal{F}_{\mathrm{E}}]$ and $G : [\mathcal{F}_{\mathrm{E}} \to \mathcal{F}_{\mathrm{E}}] \to \mathcal{F}_{\mathrm{E}}$ are defined by:

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

It is easy to check that F and G are continuous, and yield an λ-model.

THEOREM 7.4 (FILTER MODEL). *$\langle \mathcal{F}_{\mathrm{E}}, \cdot, F, G \rangle$ is a λ-model.*

Definition 7.5 (*Term interpretation*).

(1) 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:

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

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

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

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

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 Barendregt et al. [1983]. The method followed in Barendregt et al. [1983] for the proof of completeness of type assignment is to define a type interpretation υ that satisfies: for all types σ, $\upsilon(\sigma) = \{d \in \mathcal{F}_\cap \mid \sigma \in d\}$. The approach taken here is to define a function, and to show that it is a simple type interpretation.

Definition 7.7.

(1) ν_0 is defined by: $\nu_0(\sigma) = \{d \in \mathcal{F}_{\mathrm{E}} \mid \sigma \in d\}$.
(2) $\xi_\Gamma(x) = \{\sigma \in \mathcal{T} \mid \Gamma \vdash_{\mathrm{E}} x{:}\sigma\} = \uparrow_{\mathrm{E}}\tau$, where $x{:}\tau \in \Gamma$.

For ν_0 we can show:

THEOREM 7.8. *The map ν_0 is a simple type interpretation.*

We can easily show a soundness result.

THEOREM 7.9 (SOUNDNESS). $\Gamma \vdash_{\mathrm{E}} M{:}\sigma \Rightarrow \Gamma \vDash_{\mathrm{s}} M{:}\sigma$.

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

COROLLARY 7.10. *If $M =_\beta N$ and $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$, then $\Gamma \vdash_{\mathrm{E}} N{:}\sigma$.*

We can now show our completeness result.

THEOREM 7.11 (COMPLETENESS). $\Gamma \vDash_{\mathrm{s}} M{:}\sigma \Rightarrow \Gamma \vdash_{\mathrm{E}} M{:}\sigma$.

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

van Bakel [1992] also shows that strict type assignment is sound and complete with respect to inference type semantics: $\Gamma \vDash M{:}\sigma \iff \Gamma \vdash_{\mathrm{S}} M{:}\sigma$. 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 filters 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 Sections 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 7.12. Notice that

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

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 van Bakel [1992], it is shown that the map ν_0 (Definition 7.7) is an inference type interpretation.

THEOREM 7.13. *The map* $\nu_0(\sigma) = \{d \in \mathcal{F}_{\mathrm{S}} \mid \sigma \in d\}$ *is an inference type interpretation.*

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, van Bakel [1992] shows $\Gamma \vdash_{\mathrm{S}} M{:}\sigma \iff \Gamma \models M{:}\sigma$ (remark that the double turnstyle is not subscripted).

8. STRONG NORMALIZATION RESULT FOR THE SYSTEM WITHOUT ω

The other well-known result:

$$\Gamma \vdash_{\mathrm{E}} M{:}\sigma \text{ without using } \omega \iff M \text{ is strongly normalizable}$$

also holds for $\vdash_{\mathrm{E}}$, but needs a separate proof in that it is not a consequence of the Approximation Theorem 6.9. The proof for this property in van Bakel [1992] for $\vdash_{\mathrm{BCD}}$ 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 van Bakel [2008], which will be presented in Section 9.4. Alternatively, see van Bakel [2004] for a proof of this property set within $\vdash_{\mathrm{S}}$, where it is a direct consequence of the result that cut-elimination is strongly normalizable; it is this technique that will be extended to $\vdash_{\mathrm{E}}$ 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 normalizable terms. We start by defining that notion of type assignment formally.

Definition 8.1.

(1) 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 ϕ, ψ, ..., are defined through the grammar:

$$\begin{array}{rcll} \phi, \psi & ::= & \varphi \mid \sigma\to\psi & \\ \sigma, \tau & ::= & \cap_n\phi_i & (n \geq 1). \end{array}$$

We will use $\mathcal{T}_{\omega}$ 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 $\cap_n\phi_i$ rather than $n \geq 0$.)

(2) On $\mathcal{T}_{\omega}$ the relation $\leq_{\mathrm{E}}$ is as defined in Definition 4.1, except for the second alternative.

$$\begin{array}{rcll} \forall i \in \underline{n}\,[\,\cap_n\phi_i & \leq_{\mathrm{E}} & \phi_i\,] & (n \geq 1) \\ \forall i \in \underline{n}\,[\sigma \leq_{\mathrm{E}} \phi_i\,] & \Rightarrow & \sigma \leq_{\mathrm{E}} \cap_n\phi_i & (n \geq 1) \\ \rho \leq_{\mathrm{E}} \sigma \;\&\; \phi \leq_{\mathrm{E}} \psi & \Rightarrow & \sigma\to\phi \leq_{\mathrm{E}} \rho\to\psi & \end{array}$$

and the relation $\sim_{\mathrm{E}}$ is the equivalence relation generated by $\leq_{\mathrm{E}}$; the relations $\leq_{\mathrm{E}}$ and $\sim_{\mathrm{E}}$ are extended to contexts as before.

(3) We write $\Gamma \vdash_{\omega} M:\sigma$ if this statement is derivable from Γ, using only ω-free types and the derivation rules of $\vdash_{\mathrm{E}}$ (so, for rule $(\cap I)$, $n \geq 1$).

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

PROPOSITION 8.2.

$$\begin{array}{lcl}
\Gamma \vdash_{\omega} x:\sigma & \iff & \exists \rho \in \mathcal{T}\,[\,x{:}\rho \in \Gamma \ \&\ \rho \leq_{\mathrm{E}} \sigma\,] \\
\Gamma \vdash_{\omega} MN:\phi & \iff & \exists \tau\,[\,\Gamma \vdash_{\omega} M:\tau\to\phi \ \&\ \Gamma \vdash_{\omega} N:\tau\,] \\
\Gamma \vdash_{\omega} \lambda x.M:\phi & \iff & \exists \rho, \psi\,[\,\phi = \rho\to\psi \ \&\ \Gamma, x{:}\rho \vdash_{\omega} M:\psi\,] \\
\Gamma \vdash_{\omega} M:\sigma \ \&\ \Gamma' \leq_{\mathrm{E}} \Gamma \ \&\ \Gamma' \ \omega\textit{-free} & \Rightarrow & \Gamma' \vdash_{\omega} M:\sigma \\
\mathcal{D} :: \Gamma \vdash_{\omega} M:\sigma & \Rightarrow & \mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\sigma.
\end{array}$$

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

LEMMA 8.3. *If A is $\bot$-free, then there are Γ, and ϕ, such that $\Gamma \vdash_{\omega} A:\phi$.*

8.2. Strong Normalization Implies Typeability

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

LEMMA 8.4. *If $\Gamma \vdash_{\omega} M[N/x]:\sigma$ and $\Gamma \vdash_{\omega} N:\rho$, then $\Gamma \vdash_{\omega} (\lambda x.M)N:\sigma$.*

The condition $\Gamma \vdash_{\omega} N:\rho$ in the formulation of the lemma is essential and is used in part $M \equiv y \neq x$. As a counterexample, 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:

$$z{:}\sigma,\, y{:}\tau \vdash_{\omega} z:\sigma,$$

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:

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

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

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

We will now show that all strongly normalizable terms are typeable in $\vdash_{\omega}$. The proof of the crucial lemma for this result as presented in Lemma 8.6 is due to Betti Venneri, adapted to $\leq_{\mathrm{E}}$, and goes by induction on the left-most outer-most reduction path.

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

(1) there is no redex R' in M such that $\mathrm{R}' = \mathrm{C}[\mathrm{R}]$ (*outer-most*);
(2) there is no redex R' in M such that $M = \mathrm{C}_0[\mathrm{C}_1[\mathrm{R}']\,\mathrm{C}_2[\mathrm{R}]]$ (*left-most*).

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

The following lemma formulates a subject expansion result for $\vdash_{\omega}$ with respect to left-most outer-most reduction. A proof for this property in the context of $\vdash_{\mathrm{S}}$ appeared in van Bakel [2004].

LEMMA 8.6. *Let* $M \to_{\mathrm{lor}} N$, $\mathrm{lor}(M) = (\lambda x.P)Q$, $\Gamma \vdash_{\omega} N{:}\sigma$, *and* $\Gamma' \vdash_{\omega} Q{:}\tau$, *then there exists* Γ_1, ρ *such that* $\sigma \leq_{\mathrm{E}} \rho$, *and* $\Gamma_1 \vdash_{\omega} M{:}\rho$.

We can now show that all strongly normalizable terms are typeable in $\vdash_{\omega}$.

THEOREM 8.7. *If* M *is strongly normalizable, then* $\Gamma \vdash_{\omega} M{:}\sigma$ *for some* Γ, σ.

8.3. Strong Normalization

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

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

Fact 8.8

(1) If $SN(x\vec{M_i})$ and $SN(N)$, then $SN(x\vec{M_i}N)$.
(2) If $SN(M[N/x]\vec{P})$ and $SN(N)$, then $SN((\lambda x.M)N\vec{P})$.

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

$$\begin{aligned} Red(\varphi) &= \mathcal{SN} \\ Red(\sigma{\to}\phi) &= \{M \mid \forall N\,[\,N \in Red(\sigma) \Rightarrow MN \in Red(\phi)\,]\} \\ Red(\cap_n \phi_i) &= \textstyle\bigcap_{1\leq i\leq n} Red(\phi_i). \end{aligned}$$

Notice that this notion of computability is not at all defined in terms of typeability; this is the main difference between the structure of the proof here and that presented in van Bakel [1992].

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

LEMMA 8.10. *For all* ρ,

(1) $Red(\rho) \subseteq \mathcal{SN}$.
(2) $SN(x\vec{N}) \Rightarrow x\vec{N} \in Red(\rho)$.

The following result, stating that all term-variables are reducible of any type, follows immediately from Part (2):

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

We can now show that the reducibility predicate is closed for $\leq_{\mathrm{E}}$.

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

Also, the reducibility predicate is closed for subject expansion.

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

We prove our strong normalization result by showing that every typeable term is reducible. For this, we need to prove a stronger property: we 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* $\Gamma \vdash_{\omega} M{:}\sigma$, *then* $M[\overrightarrow{N_i/x_i}] \in Red(\sigma)$.

We can now prove the main result.

THEOREM 8.15 (STRONG NORMALIZATION). *Any term typeable in* $\vdash_{\omega}$ *is strongly normalizable.*

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

9. STRONG NORMALIZATION 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 normalizable, as well as that all other characterization 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 [van Bakel and Fernández 1997], which was later used for Combinator Systems in van Bakel and Fernández [2003], and was also used for $\vdash_{\mathrm{S}}$ in van Bakel [2004].

Strong normalization of cut-elimination is a well established property in the area of logic and has been profoundly studied in the past. In the area of type assignment for the λ-calculus, the corresponding property is that of strong normalization of derivation reduction (also called cut-elimination in, for example, Barendregt et al. [1983]), which mimics the reduction on terms to which the types are assigned. This area has also been well studied.

For intersection type assignment systems, proofs of strong normalization of derivation reduction have at best been indirect, for example, obtained through a mapping from the derivations into a logic by Retoré [1994], 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 [Dezani-Ciancaglini et al. 2002], but strong normalization 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 characterization 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 a normal typed- or type assignment system, not every term-redex occurs with types in a derivation.

As for the proof of this property in $\vdash_{\mathrm{S}}$ in van Bakel [2004], it is very similar to what follows, with the exception of the dealings with the contravariant $\leq_{\mathrm{E}}$ relation (Definition 9.2, and Lemmas 9.3 and 9.12).

9.1. Partial Order on Derivations

We will use the following shorthand notation for derivations.

Definition 9.1.

(1) $\mathcal{D} = \langle Ax\rangle :: \Gamma \vdash_{\mathrm{E}} x{:}\phi$ if $\mathcal{D}$ consists of nothing but an application of rule (Ax).

(2) $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I \rangle$, if and only if there are ϕ_i ($i \in \underline{n}$) such that $\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M : \phi_i$ for $i \in \underline{n}$, and $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M : \cap_n \phi_i$ is obtained from $\mathcal{D}_1, \ldots, \mathcal{D}_n$ by applying rule ($\cap I$).
(3) $\mathcal{D} = \langle \mathcal{D}', \to I \rangle$, if and only if there are M', σ, ϕ such that $\mathcal{D}' :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} M' : \phi$, and $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} \lambda x.M' : \sigma \to \phi$ is obtained from $\mathcal{D}'$ by applying rule ($\to I$).
(4) $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \to E \rangle$, if and only if there are P, Q, and σ, ϕ such that $\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} Q : \sigma$ and $\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} P : \sigma \to \phi$, and $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} PQ : \phi$ 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 subterms typed with ω, derivations involving the same term); the types derived need not be the same.

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

Definition 9.2.

(1) $\langle Ax \rangle :: \Gamma \vdash_{\mathrm{E}} x : \sigma \preceq \langle Ax \rangle :: \Gamma' \vdash_{\mathrm{E}} x : \sigma'$ for all Γ', s' with $\Gamma' \leq_{\mathrm{E}} \Gamma$, and $\sigma \leq_{\mathrm{E}} \sigma'$.
(2) $\langle \mathcal{D}_1, \ldots, \mathcal{D}_n, \cap I \rangle :: \Gamma \vdash_{\mathrm{E}} M : \cap_n \phi_i \preceq \langle \mathcal{D}'_1, \ldots, \mathcal{D}'_m, \cap I \rangle :: \Gamma' \vdash_{\mathrm{E}} M : \cap_m \phi'_j$, if and only if for every $j \in \underline{m}$ there exists an $i \in \underline{n}$ such that $\mathcal{D}_i \preceq \mathcal{D}'_j$.
(3) $\langle \mathcal{D} :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} M : \phi, \to I \rangle :: \Gamma \vdash_{\mathrm{E}} \lambda x.M : \sigma \to \phi \preceq$
$\langle \mathcal{D}' :: \Gamma', x{:}\sigma' \vdash_{\mathrm{E}} M : \phi', \to I \rangle :: \Gamma' \vdash_{\mathrm{E}} \lambda x.M' : \sigma' \to \phi'$
if and only if $\mathcal{D} \preceq \mathcal{D}'$.
(4) Let $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} P : \sigma \to \phi, \mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} Q : \sigma, \to E \rangle :: \Gamma \vdash_{\mathrm{E}} PQ : \phi$. Then, for $\sigma' \leq_{\mathrm{E}} \sigma$, $\phi' \geq_{\mathrm{E}} \phi$, $\mathcal{D}'_1 \preceq \mathcal{D}_1$, $\mathcal{D}'_2 \succeq \mathcal{D}_2$ such that $\mathcal{D}'_1 :: \Gamma' \vdash_{\mathrm{E}} P : \sigma' \to \phi'$ and $\mathcal{D}'_2 :: \Gamma' \vdash_{\mathrm{E}} Q : \sigma'$:

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

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

The following which is easy to show, generalizes Lemma 4.5, and establishes the relation between $\leq_{\mathrm{E}}$ on types and $\preceq$ on derivations.

LEMMA 9.3.

(1) If $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M : \sigma$ and $\Gamma' \leq_{\mathrm{E}} \Gamma$, $\sigma \leq_{\mathrm{E}} \sigma'$, then there exists $\mathcal{D}' \succeq \mathcal{D}$ such that $\mathcal{D}' :: \Gamma' \vdash_{\mathrm{E}} M' : \sigma'$.
(2) If $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M : \sigma \preceq \mathcal{D}' :: \Gamma' \vdash_{\mathrm{E}} M' : \sigma'$, then $\Gamma' \leq_{\mathrm{E}} \Gamma$, and $\sigma \leq_{\mathrm{E}} \sigma'$.

The first part of this proof is constructive, and, 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 contravariance 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 $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M : \sigma$. This will follow ordinary reduction, by contracting typed redexes that occur in $\mathcal{D}$: redexes for subterms of M of the shape $(\lambda x.P)Q$. The effect of this reduction will be that the derivation for the redex will be replaced by a derivation for the contractum $P[Q/x]$; this can be regarded as a generalization of cut-elimination, but has to be defined with care because the system at hand uses intersection types together with the relation $\leq_{\mathrm{E}}$.

Contracting a derivation for a redex

$$\dfrac{\dfrac{\dfrac{}{\Gamma, x{:}\sigma \vdash x : \psi}(\sigma \leq_{\mathrm{E}} \psi)}{\boxed{\mathcal{D}_1}\ \Gamma, x{:}\sigma \vdash P : \phi}(\to I) \qquad \boxed{\mathcal{D}_2}\ \Gamma \vdash Q : \sigma}{\Gamma \vdash (\lambda x.P)Q : \phi}(\to E)$$

naively gives

$$\dfrac{\boxed{\mathcal{D}_2}\ \Gamma \vdash Q : \sigma}{\Gamma \vdash Q : \psi}(\sigma \leq_{\mathrm{E}} \psi)$$
$$\boxed{\mathcal{D}_1}\ \Gamma \vdash P[Q/x] : \phi$$

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

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

(this is possible since $\mathcal{D}_2'$ exists because of Lemma 9.3, and 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 $\mathcal{D}_0 :: \Gamma \vdash_{\mathrm{E}} N : \sigma$ and $\mathcal{D} :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} M : \tau$, the derivation

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

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

(1) $\mathcal{D} = \langle Ax\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} x : \psi$, with $\sigma \leq_{\mathrm{E}} \psi$. Let $\mathcal{D}_0'$ be such that $\mathcal{D}_0 \preceq \mathcal{D}_0' :: \Gamma \vdash_{\mathrm{E}} N : \psi$, then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \mathcal{D}_0'$.
(2) $\mathcal{D} = \langle Ax\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} y : \psi$; then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle Ax\rangle :: \Gamma \vdash_{\mathrm{E}} y : \psi$.
(3) $\mathcal{D} = \langle \mathcal{D}_1, \ldots, \mathcal{D}_n,\ \cap I\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} M : \cap_n \psi_i$, 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 $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle \mathcal{D}_1', \ldots, D_n',\ \cap I\rangle :: \Gamma \vdash_{\mathrm{E}} M[N/x] : \cap_n \psi_i$.
(4) $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma, x{:}\sigma, y{:}\rho \vdash_{\mathrm{E}} M_1 : \psi, \to I\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} \lambda y.M_1 : \rho \to \psi$. Let
$$\mathcal{D}_1' = \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma, y{:}\rho \vdash_{\mathrm{E}} M_1[N/x] : \psi$$
Then $\mathcal{D}\,[\mathcal{D}_0/x{:}\sigma] = \langle \mathcal{D}_1', \to I\rangle :: \Gamma \vdash_{\mathrm{E}} (\lambda y.M_1)[N/x] : \rho \to \psi$.
(5) $\mathcal{D} = \langle \mathcal{D}_1 :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} P : \rho \to \psi, \mathcal{D}_2 :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} Q : \rho, \to E\rangle :: \Gamma, x{:}\sigma \vdash_{\mathrm{E}} PQ : \psi$. Let
$$\begin{aligned} \mathcal{D}_1' &= \mathcal{D}_1\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\mathrm{E}} P[N/x] : \rho \to \psi, \text{ and} \\ \mathcal{D}_2' &= \mathcal{D}_2\,[\mathcal{D}_0/x{:}\sigma] :: \Gamma \vdash_{\mathrm{E}} Q[N/x] : \rho, \end{aligned}$$
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 (3),

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

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:

(1) If $\mathcal{D}' = \mathcal{D}$, then $p = \varepsilon$.
(2) 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 $p = 1q$.
(3) 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 $p = 2q$.
(4) 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:

(1) $\sigma = \phi \in \mathcal{T}_{\mathrm{S}}$.
  (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:

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

  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:
    i. $\mathcal{D} = \langle \mathcal{D}_1, \to I\rangle :: \Gamma \vdash_{\mathrm{E}} \lambda x.P : \rho \to \phi$ reduces at position $1p$ with redex R to $\mathcal{D}' = \langle \mathcal{D}_1', \to I\rangle :: \Gamma \vdash_{\mathrm{E}} \lambda x.P' : \rho \to \phi$.
    ii. $\mathcal{D} = \langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle :: \Gamma \vdash_{\mathrm{E}} PQ : \phi$ 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.
    iii. $\mathcal{D} = \langle \mathcal{D}_2, \mathcal{D}_1, \to E\rangle :: \Gamma \vdash_{\mathrm{E}} PQ : \phi$ 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.
(2) $\sigma = \cap_n \phi_i$. If $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M : \cap_n \phi_i$, 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 R $= (\lambda x.P)Q$ (a subterm of M), then, for all $1 \le j \ne i \le n$, 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.
(3) 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 normalizable with respect to $\to_{\mathcal{D}}$' by '$SN(\mathcal{D})$', and use $\mathcal{SN}$ for the set of strongly normalizable 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 (1a). 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 $(\lambda x.M)N$. 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 (2) 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* $M \twoheadrightarrow_{\beta} N$.

Example 9.8. Let $\mathcal{D}_1 :: \vdash_{\mathrm{E}} \Theta:((\alpha\to\beta\to\gamma)\cap\alpha)\to((\gamma\to\delta)\cap\beta)\to\delta$,
$\mathcal{D}_2 :: \vdash_{\mathrm{E}} \Theta:\tau\to(\omega\to\phi)\to\phi$, and
$\mathcal{D} :: \vdash_{\mathrm{E}} \Theta\Theta:(\omega\to\phi)\to\phi$
be the derivations from Example 6.18, and let $\mathcal{D}'_2$ be the subderivation

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

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

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

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

We now state some standard properties of strong normalization.

LEMMA 9.9.

(1) $SN(\langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle) \Rightarrow SN(\mathcal{D}_1)$ & $SN(\mathcal{D}_2)$.
(2) $SN(\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} x\overrightarrow{M_i}:\sigma\to\phi)$ & $SN(\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} N:\sigma) \Rightarrow SN(\langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle)$.
(3) $SN(\langle \mathcal{D}_1 \cap \cdots \cap \mathcal{D}_n,\ \cap I\rangle)$ *if and only if, for all* $i \in \underline{n}$, $SN(\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i)$.
(4) *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* $SN(\mathcal{D}_3 :: \Gamma \vdash_{\mathrm{E}} \mathrm{C}[(\lambda y.M)N]:\sigma)$.

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* $\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} A:\sigma$.

9.3. Strong Normalization Result

We now come to the proof of a strong normalization result for derivation reduction. In line with the other results shown in the preceding, in order to show that each derivation is strongly normalizable with respect to $\to_{\mathcal{D}}$, a notion of computable derivations is introduced. We will show that all computable derivations are strongly normalizable 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:

$$\begin{aligned}
&Comp(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\varphi) && \iff SN(\mathcal{D})\\
&Comp(\mathcal{D}_1 :: \Gamma \vdash_{\mathrm{E}} M:\sigma\to\phi) && \iff\\
&\qquad (\,Comp(\mathcal{D}_2 :: \Gamma \vdash_{\mathrm{E}} N:\sigma) \Rightarrow Comp(\langle \mathcal{D}_1, \mathcal{D}_2, \to E\rangle :: \Gamma \vdash_{\mathrm{E}} MN:\phi))\\
&Comp(\langle \mathcal{D}_1, \ldots, \mathcal{D}_n,\ \cap I\rangle :: \Gamma \vdash_{\mathrm{E}} M:\cap_n\phi_i) && \iff \forall i \in \underline{n}\,[\,Comp(\mathcal{D}_i :: \Gamma \vdash_{\mathrm{E}} M:\phi_i)\,]
\end{aligned}$$

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 van Bakel [2004] and the one presented here lies in the fact that here we need to prove this lemma, whereas in van Bakel [2004]—where rule ($\leq_{\mathrm{S}}$) corresponds to ($\cap E$), which behavior 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* $Comp(\mathcal{D}')$.

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

LEMMA 9.13.

(*1*) $Comp(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} M:\sigma) \Rightarrow SN(\mathcal{D})$.
(*2*) $SN(\mathcal{D} :: \Gamma \vdash_{\mathrm{E}} xM_1 \cdots M_m:\sigma) \Rightarrow Comp(\mathcal{D})$.

Theorem 9.15 states 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* $Comp(\mathcal{D}[\mathcal{D}'/y:\mu] :: \Gamma \vdash_{\mathrm{E}} M[Q/y]\vec{P}:\sigma)$ *and* $Comp(\mathcal{D}' :: \Gamma \vdash_{\mathrm{E}} Q:\mu)$*, then there exists a derivation* $\mathcal{D}''$ *such that* $Comp(\mathcal{D}'' :: \Gamma \vdash_{\mathrm{E}} (\lambda y.M)Q\vec{P}:\sigma)$.

We now come to the Replacement Theorem.

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

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

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

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

9.4. New Proofs of Approximation and Strong Normalization

In this section we will show that the approximation result is a direct consequence of the strong normalization result proven in Section 9.3 for derivation reduction.

Using Theorem 9.16, we have an alternative proof for the approximation theorem:

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

Of course the proof of the characterization of (head-)normalization (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 normalizable. This result is also a consequence of the strong normalization result proven in the preceding for derivation reduction.

THEOREM 9.18. $\exists \Gamma, \sigma\,[\,\Gamma \vdash_{\omega} M:\sigma\,] \iff$ *M is strongly normalizable with respect to* $\rightarrow_\beta$.

9.5. Derivation Reduction in Other Type Assignment Systems

The previous results also hold when restricted to either $\vdash_{\mathrm{S}}$, or the relevant system $\vdash_{\mathrm{R}}$ (see Section 10.1, which corresponds to the restricted system of Definition 3.6), since these are proper subsystems of $\vdash_{\mathrm{E}}$. However, we can also show these results directly

in those other systems; we will illustrate the result of Section 9.3 by briefly looking at those systems to see that the result comes more easily there.

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

We will not discuss the proof for that result in detail here, since it would be very similar to the proof that was given in the preceding, or to that in van Bakel [2004]. The main difference lies in the fact that a relevant system is not closed for $\leq_{\mathrm{E}}$, 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 normalizable was given for $\vdash_{\mathrm{S}}$ in van Bakel [2004]. Again we will omit almost all the proof for that result here, since it would be very similar to the previous proof. 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 the following.

$(\cap E)$. Then $\mu_i = \cap_m \phi_j$ so $x_i{:}\cap_m \phi_j \in \Gamma'$, and $\sigma = \phi_k$ for some $k \in \underline{m}$. By assumption, $Comp(\mathcal{D}^i :: \Gamma \vdash_{\mathrm{S}} N_i : \cap_m \phi_j)$, and, since $\cap_m \phi_j$ is an intersection type, by definition of computability 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 $\mathcal{D}^0\,[\overrightarrow{\mathcal{D}^i/x_i{:}\mu_i}] = \mathcal{D}^i_j$, so is computable.

From these results, as in the preceding, the characterization of normalization, head-normalization, and strong normalization can be shown.

The technique used in van Bakel [1992]—which required $Comp(\Gamma, M, \sigma)$ to imply $\Gamma \vdash M : \sigma$—for the strong normalization result would not work for $\vdash_{\mathrm{S}}$, since it needs a contravariant 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 van Bakel [1995, Section 6.1]; in particular, it is needed for Lemma 6.26.2. 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 in the preceding, there exist four intersection systems for which the principal pair property is proven: a CDV-system in Coppo et al. [1980], $\vdash_{\mathrm{BCD}}$ in Ronchi Della Rocca and Venneri [1984], $\vdash_{\mathrm{S}}$ in van Bakel [1993c], and $\vdash_{\mathrm{E}}$ in van Bakel [1995], the proof of which we will repeat here.

As already discussed, 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). Next a collection of operations is identified that is proven to be sound in the sense that, when applied to a valid pair, they 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 van Bakel [1993c] (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, this operation is defined there using the contravariant 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 van Bakel [1993c] is a sound operation for $\vdash_{\mathrm{E}}$ (see Theorem 9.19): it is correct for all pairs. It is then easy to show that, with just the operations as defined in van Bakel [1993c], the principal pair property holds for $\vdash_{\mathrm{E}}$.

However, in this section a different proof will be presented, which 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 Ronchi Della Rocca and Venneri [1984], there is an overlap between operations; for example, intersections can be introduced by expansions as well as by substitutions and rise. Also, in van Bakel [1993c], 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 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 van Bakel [1993c] 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 in the preceding, that possibility existed, and especially in Coppo et al. [1980] and van Bakel [1993c], caused inconvenience, since there a "normalization-after-substitution" was called for, as explicitly defined in Coppo et al. [1980], and in part of the definition of substitution in van Bakel [1993c]. The approach of this section will be to only allow of substitutions of strict types for type variables, and to introduce a separate operation of *covering*, which deals with the assignment of ω to subterms.

In this section we will follow the scheme discussed in the preceding, when giving the proof for the principal pair property of $\vdash_{\mathrm{E}}$. For each λ-term we define the principal pair, and specify four operations on pairs of context and types, 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.

10.1. The Relevant System

The notion of *relevant* intersection type assignment is used in Damiani and Giannini [1994], van Bakel [1995], and Alessi et al. [2003], and expresses that a context can only contain those types that are actually used in a derivation—uses only types that are relevant to reach the judgement in the conclusion. Apart from not having a separate rule that deals with ω, it corresponds to the restricted system of Coppo et al. [1981] (see Definition 3.6); for convenience in proofs, we present it here as a restricted version of $\vdash_{\mathrm{E}}$.

Definition 9.1. *Relevant intersection type assignment* and *relevant intersection derivations* are defined by the following natural deduction system:

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

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

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 ($\cap I$), $\vdash_{\mathrm{R}} M{:}\omega$, for all terms M. Notice moreover, that this system is indeed truly *relevant* in the sense of Damiani and Giannini [1994]: contexts contain no more type information than that actually used in the derivation, and therefore in the ($\to I$)-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 $\psi\to\phi\to\psi$ cannot be derived, only types like $\psi\to\omega\to\psi$ can. Likewise, for $\lambda x.x$ types like $(\phi\cap\psi)\to\phi$ cannot be derived, only types like $\phi\to\phi$ 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 whether all components of that intersection type are useful for the full derivation. In this sense, $\vdash_{\mathrm{S}}$ is not relevant. Both $\vdash_{\mathrm{S}}$ and $\vdash_{\mathrm{E}}$ are conservative extensions of $\vdash_{\mathrm{R}}$, in the sense that, if $\Gamma \vdash_{\mathrm{R}} M{:}\sigma$, then also $\Gamma \vdash_{\mathrm{S}} M{:}\sigma$ and $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$.

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

LEMMA 9.2. $\exists\rho\,[\,\Gamma_1, x{:}\rho \vdash_{\mathrm{R}} M{:}\sigma \;\&\; \Gamma_2 \vdash_{\mathrm{R}} N{:}\rho\,] \Longleftrightarrow \bigcap\{\Gamma_1, \Gamma_2\} \vdash_{\mathrm{R}} M[N/x]{:}\sigma$.

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

As for $\vdash_{\mathrm{BCD}}$ (Theorem 6.12), the relation between $\vdash_{\mathrm{E}}$ and $\vdash_{\mathrm{R}}$ is formulated for terms in $\mathcal{A}$ by:

LEMMA 9.3. *For all $A \in \mathcal{A}$: if $\Gamma \vdash_{\mathrm{E}} A{:}\sigma$, then there are Γ', σ' such that $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$, $\sigma' \leq_{\mathrm{E}} \sigma$ and $\Gamma \leq_{\mathrm{E}} \Gamma'$.*

Using the same technique as in Section 6.1, the following theorem can be proven.

THEOREM 9.4. $\Gamma \vdash_{\mathrm{R}} M{:}\sigma \Longleftrightarrow \exists A \in \mathcal{A}(M)\,[\,\Gamma \vdash_{\mathrm{R}} A{:}\sigma]$.

Using this approximation result for $\vdash_{\mathrm{R}}$, the following becomes easy.

THEOREM 9.5. *If $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$, then there are $\Gamma' \geq_{\mathrm{E}} \Gamma$, $\sigma' \leq_{\mathrm{E}} \sigma$ such that $\Gamma' \vdash_{\mathrm{R}} M{:}\sigma$.*

10.2. Principal Pairs

Principal pairs for both $\vdash_{\mathrm{R}}$ and $\vdash_{\mathrm{E}}$ are defined by:

Definition 9.6 (*Principal Pairs*).

(1) Let $A \in \mathcal{A}$. $pp(A)$, the *principal pair* of A, is defined by:
   (a) $pp(\bot) = \langle \emptyset, \omega\rangle$.
   (b) $pp(x) = \langle \{x{:}\varphi\}, \varphi\rangle$.
   (c) If $A \neq \bot$, and $pp(A) = \langle \Pi, \pi\rangle$, then:
      i. If x occurs free in A, and $x{:}\sigma \in \Pi$, then $pp(\lambda x.A) = \langle \Pi\backslash x, \sigma\to\pi\rangle$.
      ii. Otherwise $pp(\lambda x.A) = \langle \Pi, \omega\to\pi\rangle$.
   (d) If for $i \in \underline{n}$, $pp(A_i) = \langle \Pi_i, \pi_i\rangle$ (disjoint in pairs), then
$$pp(xA_1\cdots A_n) = \langle \textstyle\bigcap_n \Pi_i \cap \{x{:}\pi_1\to\cdots\to\pi_n\to\varphi\}, \varphi\rangle,$$
   where φ is a type-variable that does not occur in $pp(A_i)$, for $i \in \underline{n}$.
(2) $\mathcal{P} = \{\langle \Pi, \pi\rangle \mid \exists A \in \mathcal{A}\,[pp(A) = \langle \Pi, \pi\rangle\,]\}$.

The principal pairs in the systems as presented in Coppo et al. [1980], $\vdash_{\mathrm{BCD}}$ in Ronchi Della Rocca and Venneri [1984], and $\vdash_{\mathrm{S}}$ in van Bakel [1993c] are exactly as in the preceding. Since the essential type assignment system is a subsystem of $\vdash_{\mathrm{BCD}}$, and it is a supersystem $\vdash_{\mathrm{S}}$, which in turn is a supersystem of $\vdash_{\mathrm{R}}$, this is not surprising.

The following result is almost immediate:

LEMMA 9.7. *If* $pp(A) = \langle \Pi, \pi \rangle$, *then* $\Pi \vdash_{\mathrm{R}} A{:}\pi$.

The notion of principal pairs for terms in $\mathcal{A}$ will be generalized to arbitrary λ-terms in Definition 9.24.

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 Ronchi Della Rocca and Venneri [1984], this kind of operation will proven to be sufficient.

Definition 9.8.

(1) 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:

$$\begin{array}{lll} (\varphi \mapsto \psi)(\varphi) & = & \psi \\ (\varphi \mapsto \psi)(\varphi') & = & \varphi', \qquad \textit{if } \varphi \not\equiv \varphi' \\ (\varphi \mapsto \psi)(\sigma \to \phi) & = & (\varphi \mapsto \psi)(\sigma) \to (\varphi \mapsto \psi)(\phi) \\ (\varphi \mapsto \psi)(\cap_n \phi_i) & = & \cap_n (\varphi \mapsto \psi)(\phi_i) \end{array}$$

(2) If S_1 and S_2 are substitutions, then so is $S_1 \circ S_2$, where $S_1 \circ S_2(\sigma) = S_1(S_2(\sigma))$.
(3) $S(\Gamma) = \{x{:}S(\alpha) \mid x{:}\alpha \in \Gamma\}$.
(4) $S(\langle \Gamma, \sigma \rangle) = \langle S(\Gamma), S(\sigma) \rangle$.

The operation of substitution is sound for $\vdash_{\mathrm{R}}$.

THEOREM 9.9 (SOUNDNESS OF SUBSTITUTION). *If* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$, *then for every substitution* S: *if* $S(\langle \Gamma, \sigma \rangle) = \langle \Gamma', \sigma' \rangle$, *then* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$.

10.4. Expansion

The operation of expansion on types defined here corresponds to the notion of expansion as defined for $\vdash_{\mathrm{BCD}}$ in Ronchi Della Rocca and Venneri [1984] and for $\vdash_{\mathrm{S}}$ in van Bakel [1993c]. A difference between the notions of expansion as defined in Coppo et al. [1980] and Ronchi Della Rocca and Venneri [1984] 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 9.10.

(1) 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 $\langle \Gamma, \sigma \rangle$.
(2) The *last type-variable* of a strict type ϕ, *last-tv*(ϕ), is defined by:

$$\begin{array}{rcl} \textit{last-tv}(\varphi) & = & \varphi \\ \textit{last-tv}(\sigma \to \varphi) & = & \textit{last-tv}(\varphi) \end{array}$$

An expansion is essentially an operation on derivations characterized 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 9.11. For every ϕ, $n \geq 2$, context Γ, and σ, the quadruple $\langle \phi, n, \Gamma, \sigma \rangle$ determines an *expansion* $Exp_{\langle \phi, n, \Gamma, \sigma \rangle} : \mathcal{T} \to \mathcal{T}$, that is constructed as follows.

(1) 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 $\psi \in \mathcal{T}_S\langle\Gamma,\sigma\rangle$, and $\mathit{last\text{-}tv}(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, then for all type-variables φ that occur in ψ: $\varphi \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$.

(2) 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 $j \in \underline{m}$.

(3) $\mathit{Exp}_{\langle\phi,n,\Gamma,\sigma\rangle}(\tau)$ is obtained by traversing τ top-down and replacing every subtype ϕ by $S_1(\phi) \cap \cdots \cap S_n(\phi)$, if $\mathit{last\text{-}tv}(\phi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle$, i.e. (where $\mathit{Ex} = \mathit{Exp}_{\langle\phi,n,\Gamma,\sigma\rangle}$):

$$\begin{array}{lcll} \mathit{Ex}(\cap_n\psi_i) & = & \cap_n(\mathit{Ex}(\psi_i)) & \\ \mathit{Ex}(\psi) & = & \cap_n(S_i(\psi)) & \mathit{if\ last\text{-}tv}(\psi) \in \mathcal{V}_\phi\langle\Gamma,\sigma\rangle \\ \mathit{Ex}(\rho\to\psi) & = & \mathit{Ex}(\rho)\to\mathit{Ex}(\psi) & \mathit{if\ last\text{-}tv}(\psi) \notin \mathcal{V}_\phi\langle\Gamma,\sigma\rangle \\ \mathit{Ex}(\varphi) & = & \varphi & \mathit{if}\ \varphi \notin \mathcal{V}_\phi\langle\Gamma,\sigma\rangle \end{array}$$

(4) $\mathit{Ex}(\Gamma') = \{x{:}\mathit{Ex}(\rho) \mid x{:}\rho \in \Gamma'\}$.
(5) $\mathit{Ex}(\langle\Gamma',\sigma'\rangle) = \langle\mathit{Ex}(\Gamma'),\mathit{Ex}(\sigma')\rangle$.

Instead of $\mathit{Exp}_{\langle\phi,n,\Gamma,\sigma\rangle}$, we will write $\langle\phi,n,\Gamma,\sigma\rangle$.

Example 9.12. 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

$$\mathit{Ex}(\phi) = ((\varphi_4\cap\varphi_5)\to\varphi_2)\to(\varphi_6\to\varphi_4)\cap(\varphi_7\to\varphi_5)\to\varphi_6\cap\varphi_7\to\varphi_2.$$

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

THEOREM 9.13 (SOUNDNESS OF EXPANSION). *If* $\Gamma \vdash_R A{:}\sigma$ *and let* Ex *be such that* $\mathit{Ex}(\langle\Gamma,\sigma\rangle) = \langle\Gamma',\sigma'\rangle$, *then* $\Gamma \vdash_R A{:}\sigma$.

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 9.23).

Definition 9.14. The relation on pairs $\lessdot$ is defined by:

$$\begin{array}{rcll} \langle\Gamma,\sigma\rangle & \lessdot & \langle\emptyset,\omega\rangle & \\ \langle\bigcap_n\Gamma_i,\cap_n\phi_i\rangle & \lessdot & \langle\bigcap_n\Gamma_i',\cap_n\phi_i'\rangle & \mathit{iff}\ \forall i \in \underline{n}\ (n \geq 2)\ [\,\langle\Gamma_i,\phi_i\rangle \lessdot \langle\Gamma_i',\phi_i'\rangle\,] \\ \langle\Gamma,\rho\to\psi\rangle & \lessdot & \langle\Gamma',\rho'\to\psi'\rangle & \mathit{iff}\ \langle\Gamma\cup\{x{:}\rho\},\psi\rangle \lessdot \langle\Gamma'\cup\{x{:}\rho'\},\psi'\rangle \\ \langle\bigcap_n\Gamma_i\cap\{x{:}\sigma_1\to\cdots\to\sigma_n\to\sigma\},\sigma\rangle & \lessdot & \langle\bigcap_n\Gamma_i'\cap\{x{:}\sigma_1'\to\cdots\to\sigma_n'\to\sigma\},\sigma\rangle & \\ & & & \mathit{iff}\ \forall i \in \underline{n}\ [\,\langle\Gamma_i,\sigma_i\rangle \lessdot \langle\Gamma_i',\sigma_i'\rangle\,] \end{array}$$

Definition 9.15. 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 $\langle\Gamma_0,\tau_0\rangle \lessdot \langle\Gamma_1,\tau_1\rangle$, and is defined by:

$$\begin{array}{rcll} \mathit{Cov}(\langle\Gamma,\sigma\rangle) & = & \langle\Gamma_1,\tau_1\rangle, & \mathit{if}\ \langle\Gamma,\sigma\rangle = \langle\Gamma_0,\tau_0\rangle, \\ & = & \langle\Gamma,\sigma\rangle, & \mathit{otherwise}. \end{array}$$

The operation of covering is not sound as a general operation for $\vdash_E$.

Example 9.16. It is easy to check that $\langle\{x{:}\alpha\cap(\alpha\to\alpha)\},\alpha\rangle \lessdot \langle\{x{:}\omega\to\alpha\},\alpha\rangle$. Notice that the first pair is legal for x since $x{:}\alpha\cap(\alpha\to\alpha) \vdash_E x{:}\alpha$ is derivable, but we cannot derive $x{:}\omega\to\alpha \vdash_E x{:}\alpha$.

The operation of covering is however, sound for $\vdash_R$.

THEOREM 9.17 (SOUNDNESS OF COVERING). *If* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$*, and Cov is a covering, such that* $Cov(\langle \Gamma, \sigma \rangle) = \langle \Gamma', \sigma' \rangle$*, then* $\Gamma' \vdash_{\mathrm{R}} A{:}\sigma'$.

10.6. Lifting

The last operation needed in this section is that of lifting.

Definition 9.18.
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:

(1) $L(\sigma) = \tau_1$, if $\sigma = \tau_0$; otherwise, $L(\sigma) = \sigma$.
(2) $L(\Gamma) = \Gamma_1$, if $\Gamma = \Gamma_0$; otherwise, $L(\Gamma) = \Gamma$.
(3) $L(\langle \Gamma, \sigma \rangle) = \langle L(\Gamma), L(\sigma) \rangle$.

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 $\vdash_{\mathrm{R}} \lambda x.x{:}\phi \cap \psi \to \phi$.
But we can show that the operation of lifting is sound for $\vdash_{\mathrm{E}}$.

THEOREM 9.19 (SOUNDNESS OF LIFTING). *If* $\Gamma \vdash_{\mathrm{E}} M{:}\sigma$ *and* $\langle \langle \Gamma, \sigma \rangle, \langle \Gamma', \sigma' \rangle \rangle$ *is a lifting, then* $\Gamma' \vdash_{\mathrm{E}} M{:}\sigma'$.

10.7. Completeness of Operations

We will now show completeness of these specified operations, both for $\vdash_{\mathrm{R}}$ as for $\vdash_{\mathrm{E}}$. First the notion of chain of operations is introduced.

Definition 9.20.

(1) A *chain of operations* is an object $[O_1, \ldots, O_n]$, where each O_i is an expansion, covering, substitution, or lifting, and
$$[O_1, \ldots, O_n](\langle \Gamma, \sigma \rangle) = O_n(\cdots(O_1(\langle \Gamma, \sigma \rangle))\cdots).$$
(2) On chains the operation of concatenation is denoted by $*$, and
$$[O_1, \ldots, O_i] * [O_{i+1}, \ldots, O_n] = [O_1, \ldots, O_n].$$
(3) 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: $\overrightarrow{Ex_i} * [S, Cov]$.
(4) 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.

THEOREM 9.21 (COMPLETENESS OF RELEVANT CHAINS). *If* $\Gamma \vdash_{\mathrm{R}} A{:}\sigma$ *and* $pp(A) = \langle \Pi, \pi \rangle$*, then there exists a relevant chain Ch such that* $Ch(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.

We can now show a completeness result for $\vdash_{\mathrm{E}}$:

THEOREM 9.22 (COMPLETENESS OF ESSENTIAL CHAINS). *If* $\Gamma \vdash_{\mathrm{E}} A{:}\sigma$ *and* $pp(A) = \langle \Pi, \pi \rangle$*, then there exists an essential chain Ch such that* $Ch(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.

As in Coppo et al. [1981], Ronchi Della Rocca and Venneri [1984], and van Bakel [1993c], 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 defined in Ronchi Della Rocca and Venneri [1984] 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 $\lessdot$ on pairs as given in Definition 9.14 is used.

THEOREM 9.23. $\langle \mathcal{P}, \gg \rangle$ *is a meet semi-lattice isomorphic to* $\langle \mathcal{A}, \sqsubseteq \rangle$.

Definition 9.24.

(1) Let M be a term. Let $\mathcal{P}(M)$ be the set of all principal pairs for all approximants of M: $\mathcal{P}(M) = \{pp(A) \mid A \in \mathcal{A}(M)\}$.
(2) $\mathcal{P}(M)$ 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 9.25. *Let* Γ *and* σ *be such that* $\Gamma \vdash_{\mathrm{E}} M : \sigma$.

(1) $\mathcal{A}(M)$ *is finite. Let* $\langle \Pi, \pi \rangle$ *be the principal pair of* M. *Then there exists an essential chain* Ch *such that* $Ch(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.
(2) $\mathcal{A}(M)$ *is infinite. Then there exist a pair* $\langle \Pi, \pi \rangle \in \mathcal{P}(M)$ *and an essential chain* Ch *such that* $Ch(\langle \Pi, \pi \rangle) = \langle \Gamma, \sigma \rangle$.

The same result, using relevant chains rather than essential chains, holds 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 (an infinite Böhm-tree) have 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 1978 to characterize normalization, the concept quickly evolved into the well-known system defined by Barendregt, Coppo, and Dezani in 1983. This was shown to have many strong properties, including giving rise to filter semantics and completeness, and characterizing head-normalization and normalization. Other important elementary properties, like the approximation result and the principal pairs property were later shown by Ronchi della Rocca and Venneri in 1984.

A few years later, in 1988 I myself discovered not only the strength of intersection types, but also their beauty and elegance, especially when proving the strong normalization 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 1980 and 1981—proved to be as expressive as the BCD-system, in that in 1988 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. In 1991 I showed that this strict system has the principal pair property in, where the lack of extensionality of the system created complications in the proofs.

In 1992, I noticed that by adding contravariance 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 characterization and termination results, as well as the principal pair property. In 1995, in order to show the characterization of head-normalization, and normalization, I first showed the approximation result.

After having proven a number of characterization and termination results using Tait's technique, I could not help but observe that these proofs had much structure in common, and looked for a common factor; this turned out to be the strong normalization of derivation reduction. In 2001 I showed this for the strict system, and showed that all characterization results follow from this main result; in 2005 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.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

ALESSI, F., BARBANERA, F., AND DEZANI-CIANCAGLINI, M. 2003. Intersection types and computational rules. *Electro. Notes Theoret. Comput. Sci. 84*.

BAKEL, S. VAN 1988. Derivations in type assignment systems. M.S. thesis, University of Nijmegen.

BAKEL, S. VAN 1992. Complete restrictions of the intersection type discipline. *Theoret. Comput. Sci. 102,* 1, 135–163.

BAKEL, S. VAN 1993a. Essential intersection type assignment. In *Proceedings of 13th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS)*. R. Shyamasunda, Ed. Lecture Notes in Computer Science, vol. 761. Springer-Verlag, 13–23.

BAKEL, S. VAN 1993b. Intersection type disciplines in lambda calculus and applicative term rewriting systems. Ph.D. thesis, Department of Computer Science, University of Nijmegen.

BAKEL, S. VAN 1993c. Principal type schemes for the strict type assignment system. *J. Logic Computat. 3,* 6, 643–670.

BAKEL, S. VAN 1995. Intersection type assignment systems. *Theoret. Comput. Sci. 151,* 2, 385–435.

BAKEL, S. VAN 1996. Rank 2 intersection type assignment in term rewriting systems. *Fundamenta Informaticae 2,* 26, 141–166.

BAKEL, S. VAN 2002. Rank 2 types for term graph rewriting (extended abstract). In *Electronic Proceedings of the International Workshop on Types in Programming (TIP)*. Dagstuhl, Germany. Vol. 75.

BAKEL, S. VAN 2004. Cut-elimination in the strict intersection type assignment system is strongly normalising. *Notre Dame J. Formal Logic 45,* 1, 35–63.

BAKEL, S. VAN 2008. The heart of intersection type assignment; Normalization proofs revisited. *Theoret. Comput. Sci. 398*, 82–94.

BAKEL, S. VAN AND FERNÁNDEZ, M. 1997. Normalization Results for Typeable Rewrite Systems. *Inform. Comput. 2,* 133, 73–116.

BAKEL, S. VAN AND FERNÁNDEZ, M. 2003. Normalisation, Approximation, and Semantics for Combinator Systems. *Theoret. Comput. Sci. 290*, 975–1019.

BARENDREGT, H. 1984. *The Lambda Calculus: Its Syntax and Semantics.* Revised Ed. North-Holland, Amsterdam.

BARENDREGT, H., COPPO, M., AND DEZANI-CIANCAGLINI, M. 1983. A filter lambda model and the completeness of type assignment. *J. Symbol. Logic 48,* 4, 931–940.

BENTON, P. N. 1993. Strictness analysis of lazy functional programs. Tech. rep. 309 Computing Lab, University of Cambridge.

CARDELLI, L. 1987. Basic polymorphic typechecking. *Sci. Comput. Program. 8,* 2, 147–172.

CHURCH, A. 1936. A note on the entscheidungsproblem. *J. Symbol. Logic 1,* 1, 40–41.

COPPO, M., DEZANI, M., AND SALLÉ, P. 1979. Functional characterization of some semantic equalities inside λ-calculus. In *Proceedings of* 6th *International Colloquium on Automata, Languages and Programming (ICALP)*. H. Maurer, Ed., Lecture Notes in Computer Science, vol. 71, Springer-Verlag, 133–146.

COPPO, M. AND DEZANI-CIANCAGLINI, M. 1980. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic 21,* 4, 685–693.

COPPO, M., DEZANI-CIANCAGLINI, M., AND VENNERI, B. 1980. Principal type schemes and λ-calculus semantics. In *To H.B. Curry, Essays in Combinatory Logic, Lambda-Calculus and Formalism*, J. Hindley and J. Seldin, Eds., Academic Press, New York, 535–560.

COPPO, M., DEZANI-CIANCAGLINI, M., AND VENNERI, B. 1981. Functional characters of solvable terms. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 27*, 45–58.

COPPO, M., DEZANI-CIANCAGLINI, M., AND ZACCHI, M. 1987. Type theories, normal forms and D_∞-lambda-models. *Inform. Comput. 72,* 2, 85–116.

COPPO, M. AND FERRARI, A. 1993. Type inference, abstract interpretation and strictness analysis. In *A Collection of Contributions in Honour of Corrado Böhm*, M. Dezani-Ciancaglini, S. Ronchi Della Rocca, and M. Venturini Zilli, Eds., Elsevier, 113–145.

CURRY, H. 1934. Functionality in combinatory logic. *Proc. Nat. Acad. Sci. 20*, 584–590.

CURRY, H. AND FEYS, R. 1958. *Combinatory Logic*. Vol. 1. North-Holland, Amsterdam.

DAMAS, L. AND MILNER, R. 1982. Principal type-schemes for functional programs. In *Proceedings of the 9*th *ACM Symposium on Principles of Programming Languages*. 207–212.

DAMIANI, F. 2000. Typing local definitions and conditional expressions with Rank 2 intersection. In *Proceedings of (FOSSACS)*. Lecture Notes in Computer Science, vol. 1784. Springer-Verlag, 82–97.

DAMIANI, F. 2003. Rank 2 intersection types for local definitions and conditional expressions. *ACM Trans. Program. Lang. Syst. 25,* 4, 401–451.

DAMIANI, F. AND GIANNINI, P. 1994. A decidable intersection type system based on relevance. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Software,(TACS)*. M. Hagiya and J. Mitchell, Eds., Lecture Notes in Computer Science, vol. 789, Springer-Verlag, 707–725.

DEZANI-CIANCAGLINI, M. AND MARGARIA, I. 1986. A characterisation of F-complete type assignments. *Theoret. Comput. Sci. 45*, 121–157.

DEZANI-CIANCAGLINI, M., MEYER, R., AND MOTOHAMA, Y. 2002. The semantics of entailment omega. *Notre Dame j. Formal Logic 43,* 3, 129–145.

ENGELER, E. 1981. Algebras and combinators. *Algebra Universalis 13,* 3, 389–392.

GIRARD, J. 1986. The system f of variable types, fifteen years later. *Theoret. Comput. Sci. 45*, 159–192.

GUNTER, C. AND SCOTT, D. 1990. Semantic domains. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., North-Holland, 633–674.

HINDLEY, J. 1969. The principal type scheme of an object in combinatory logic. *Trans. Amer. Math. Soc. 146*, 29–60.

HINDLEY, J. 1982. The simple semantics for Coppo-Dezani-Sallé type assignment. In *Proceedings of the International Symposium on Programming*. M. Dezani and U. Montanari, Eds., Lecture Notes in Computer Science, vol. 137. Springer-Verlag, 212–226.

HINDLEY, J. 1983. The completeness theorem for typing λ-terms. *Theoret. Comput. Sci. 22,* 1, 1–17.

HINDLEY, J. 1997. *Basic Simple Type Theory*. Cambridge University Press.

HINDLEY, R. AND LONGO, G. 1980. Lambda calculus models and extensionality. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik 26*, 289–310.

HUDAK, P., PEYTON JONES, S., WADLER, P., BOUTEL, B., FAIRBAIRN, J., FASEL, J., HAMMOND, K., HUGHES, J., JOHNSSON, T., KIEBURTZ, D., NIKHIL, R., PARTAIN, W., AND PETERSON, J. 1992. Report on the programming language Haskell. *ACM SIGPLAN Not. 27,* 5, 1–64.

JENSEN, T. 1992. Abstract interpretation in logical form. Ph.D. thesis, Imperial College, University of London.

JENSEN, T. P. 1995. Conjunctive type systems and abstract interpretation of higher-order functional programs. *J. Logic Comput. 5,* 4, 397–421.

JIM, T. 1996. What are principal typings and what are they good for? In *Proceedings of the 23rd ACM Symposium on Principles of Programming Languages (POPL)*. 42–53.

KFOURY, A., MAIRSON, H., TURBAK, F., AND WELLS, J. 1999. Relating typability and expressibility in finite-rank intersection type systems. In *Proceedings of the International Conference on Functional Programming (CFP)*. 90–101.

KFOURY, A. AND WELLS, J. 1999. Principality and decidable type inference for finite-rank intersection types. In *Proceedings of the 26th ACM Symposium on the Principles of Programming Languages (POPL)*. 161–174.

LEIVANT, D. 1983. Polymorphic type inference. In *Proceedings of the 10*th *ACM Symposium on Principles of Programming Languages*. 88–98.

MILNER, R. 1978. A theory of type polymorphism in programming. *J. Comput. Syst. Sci. 17*, 348–375.

MITCHELL, J. 1988. Polymorphic type inference and containment. *Inform. Comput. 76*, 211–249.

PIERCE, B. 1991. Programming with intersection types and bounded polymorphism. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA.

RETORÉ, C. 1994. A note on intersection types. INRIA Rapport de recherche 2431, INRIA, France.

REYNOLDS, J. C. 1981. The essence of Algol. In *Algorithmic Languages*, J. de Bakker and J. van Vliet, Eds., North-Holland, 345–372.

REYNOLDS, J. C. 1988. Preliminary design of the programming language Forsythe. Tech. rep. CMU-CS-88-159, Carnegie Mellon University, Pitssburgh, PA.

ROBINSON, J. A. 1965. A machine-oriented logic based on resolution principle. *J. ACM 12,* 1, 23–41.

RONCHI DELLA ROCCA, S. 1988. Principal type scheme and unification for intersection type discipline. *Theoret. Comput. Sci. 59*, 181–209.

RONCHI DELLA ROCCA, S. AND VENNERI, B. 1984. Principal type schemes for an extended type theory. *Theoret. Comput. Sci. 28*, 151–169.

SALLÉ, P. 1978. Une extension de la théorie des types. In *Proceedings of the 5th Colloquium on Automata, Languages and Programming*. G. Ausiello and C. Böhm, Eds., Lecture Notes in Computer Science, vol. 62, Springer-Verlag, 398–410.

TAIT, W. 1967. Intensional interpretation of functionals of finite type I. *J. Symbol. Logic 32,* 2, 198–223.

TERAUCHI, T. AND AIKEN, A. 2006. On typability for rank-2 intersection types with polymorphic recursion. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*. 111–122.

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

WELLS, J. 2002. The essence of principal typings. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*. Lecture Notes in Computer Science, vol. 2380. Springer-Verlag, 913–925.

Received April 2006; revised January 2009, June 2009; accepted October 2009