Intersection Types for λ-Trees

Steffen van Bakel [a,⋆] Franco Barbanera [b,⋆,⋆⋆]
Mariangiola Dezani-Ciancaglini [c,⋆,⋆⋆] Fer-Jan de Vries [d]

[a] *Department of Computing, Imperial College,*
180 Queen's Gate, London SW7 2BZ, U.K.
E-mail: svb@doc.ic.ac.uk

[b] *Dipartimento di Matematica e Informatica, Università degli Studi di Catania,*
Viale A. Doria 6, 95125 Catania, Italia.
E-mail: barba@dmi.unict.it.

[c] *Dipartimento di Informatica, Università degli Studi di Torino*
Corso Svizzera 185, 10149 Torino, Italia,
Email: dezani@di.unito.it

[d] *Computer Science Division, Electrotechnical Laboratory,*
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan,
E-mail: ferjan@etl.go.jp

Abstract

We introduce a type assignment system which is parametric with respect to five families of trees obtained by evaluating λ-terms (Böhm trees, Lévy-Longo trees, ...). Then we prove, in an (almost) uniform way, that each type assignment system fully describes the observational equivalences induced by the corresponding tree representation of λ-terms. More precisely, for each family of trees, two λ-terms have the same tree if and only if they get assigned the same types in the corresponding type assignment system.

Key words: Böhm trees, approximants, intersection types.

1 Introduction

A theory of functions like the λ-calculus, which provides a foundation for the functional programming paradigm in computer science, can be seen, essentially, as a theory of 'programs'. This point of view leads naturally to the intuitive idea that

⋆ Partly supported by NATO Grant CR.G.970285.
⋆⋆ Partly supported by MURST COFIN'99 TOSCA Project.

the meaning of a λ-term (program) is represented by the amount of 'meaningful information' we can extract from that λ-term by 'running it'. The formalization of 'the information' obtained from a λ-term requires, first, the definition of what is, in a λ-term, a 'stable relevant minimal information' that is directly observable in the λ-term. This is the token of information which cannot be altered by further reductions but can only be added upon. (As an example, the reader may think of the calculation of $\sqrt{2}$. The calculation process merely adds decimals to the already calculated decimal expansion).

If one organizes the stable relevant minimal information produced during a computation according to the order in which it is obtained, it is quite natural to get a tree representation of the information implicitly contained in the original λ-term. This tree then embodies the total information hidden in the original λ-term. There are many such tree representations in literature, depending on the possible notions of stable relevant minimal information; the most commonly used being *top* trees (or Berarducci trees [6]), *weak* trees (or Lévy-Longo trees [25]), *head* trees (or Böhm trees [4]), *eta* trees and *infinite eta* trees (*infinite eta* trees are in one-one correspondence with Nakajima trees [23]). Hence, the various notions of tree represent different notions of *meaning of a λ-term* (in particular, they specify different notions of *undefined* value [20]).

This apparently vague intuition is substantiated by results starting with [29], which show that there exist precise correspondences between the tree representations of λ-terms and the local structures (or, equivalently, the λ-theories) of certain λ-models ([4], Chapter 19). In particular, such correspondences amount to the fact that two λ-terms have the same tree representation if and only if they are equal in the λ-model. For example,

- the *infinite eta* trees represent the local structure of Scott's D_∞ model as defined in [26] (this result was proved in [29]);
- the *eta* trees represent the local structure of the inverse limit model defined in [12];
- the *head* trees represent the local structure of Scott's P_ω model as defined in [27] (a discussion on this topic can be found in [4], Chapter 19);
- the *weak* trees were introduced by Longo in [22] (following [21]), who proved that they represent the local structure of Engeler's models as defined in [17].

Orthogonally, the results about observational equivalences confirm this operational intuition of *dynamically* evolving meanings of λ-terms incorporated in the tree representations. For instance, in [29] Wadsworth showed that two λ-terms M, N have the same *infinite eta* tree if and only if, for all contexts $C[\,]$, the following holds:

$C[M]$ has a head normal form if and only if $C[N]$ has a head normal form.

The same property holds even considering *eta* trees and normal forms [18]. By adding a non-deterministic choice operator and an adequate numeral system to the pure calculus, we obtain a language which internally discriminates two λ-terms if and only if they have different *head* trees [14]. *Weak* trees correspond to the observational equivalence with respect to weak head normal forms in suitably enriched versions of the λ-calculus [25,9,16]. We can discriminate λ-terms in the same way that top trees do, using two powerful δ-rules [15].

It is clear that most of the relevant properties of λ-terms pertain, more or less strongly, to the field of *dynamics*, i.e., to their computational behavior. This, however, does not mean that we have to, staying into a 'physics metaphor', disregard the *statics*: the objects of a theory of programs (before we 'run' them), are static entities and, as such, differently from the more or less ineffable *computations*, they can be 'handled'.

It would be very useful if these dynamic aspects could be analyzed with tools dealing with static entities like, for instance, λ*-terms* and *types*.

All the results recalled above show that our dynamic world can be partially reduced to a world of trees. Trees are objects a bit more concrete than *computations*, but still not very manageable. Type assignment disciplines are typical static tools, much used in the programming practice to check decidable properties of programs. There are several results showing how very powerful typing disciplines can be devised that, at the (of course expected) price of being undecidable, can be used to analyze the dynamic world. For instance, the observational equivalences induced by a number of tree representations of λ-terms can be mimicked by suitable type theories:

- Each inverse limit λ-model is isomorphic to a filter model, i.e., to a model in which the meaning of λ-terms is a set of derivable intersection types [10].
- Two λ-terms have the same *head* tree if and only if they have the same set of types in the standard intersection type discipline [5], as proved in [24].
- Two λ-terms have the same *weak* tree if and only if they have the same set of types in the type discipline with union and intersection of [13], as proved in [16].
- Two λ-terms have the same *top* tree if and only if they have the same set of types in a type assignment system with applicative types [7].

In the present paper we will design one type assignment system for each of the five families of trees mentioned above (more precisely, a type assignment system (almost) parametric with respect to these five families). For each family of trees we will show that two λ-terms have the same tree, if and only if they get assigned the same types in the corresponding type assignment system.

This is a new result for the *eta* trees and the *infinite eta* trees. Moreover, our proof method unifies the earlier proofs mentioned above, while making the following improvements:

- we simplify the types of [24], since we do not consider type variables;

- we do not allow the union type constructor (which is considered in [16]);
- the applicative types are built starting from just one constant instead of two (this was the choice of [7]).

All the type systems we will introduce (apart from those that represent *top* trees) induce filter λ-models in the sense of [5]. Clearly, the theories of these filter models coincide with the equalities of the corresponding trees. So as by-product we obtain alternative proofs of the characterizations of the theories of Scott's D_∞ model [29] and of the filter λ-model [24]. Notice that these new proofs (unlike the original ones) are constructive, in the sense that, whenever two λ-terms have different interpretations, we will build a compact element d of the model such that d approximates only the interpretation of one of the two λ-terms. Indeed, d is the principal filter induced by a type which can be deduced only for one of the two λ-terms.

The long-term goal of this research is to find answers to the question 'what can be added to the pure λ-calculus in order to internally discriminate λ-terms having different trees?', which can be formulated for each family of trees.

Intersection type assignment systems played a crucial role in showing that observational equivalences in suitable extensions of λ-calculus are equivalent to *head* and *weak* tree equality [9,14,16]. We hope that similar results can be obtained for the other families of trees; this would justify the present choices. A very limited number of type constants and type constructors allows to search for a proof along the following lines. Suppose we were able to define, for each type α, a test term $\mathbf{T}_\alpha$ such that $\mathbf{T}_\alpha M$ converges if and only if M has type α. Then we would obtain an observational equivalence which coincides with the tree equality (see [8]).

This paper is organized as follows. In Section 2 we shall recall the various definitions of tree. We will introduce the notion of *approximant* in Section 3. In Section 4 we will describe the type assignment systems which will be used for our main result and we will give a theorem of approximation stating that a λ-term has a type if and only if there exists an approximant of the λ-term which has the same type. Section 5, instead, contains our main result: our type assignment systems can be used to analyze the observational behavior represented by trees.

A preliminary version of this paper has appeared in [3], where almost all proofs were omitted.

Abbreviations

Below, we will use the following abbreviations for λ-terms.

$$\begin{array}{llll} \mathbf{Y} & \equiv \lambda f.f((\lambda x.f(xx))(\lambda x.f(xx))) & \mathbf{R} & \equiv \lambda zxy.x(zzy) \\ \mathbf{I} & \equiv \lambda x.x & \mathbf{\Delta_n} & \equiv \lambda x.\overbrace{x\ldots x}^{n} \\ \mathbf{\Omega_n} & \equiv \overbrace{\mathbf{\Delta_n}\ldots\mathbf{\Delta_n}}^{n} & \mathbf{\Delta_2^\eta} & \equiv \lambda xy.xxy. \end{array}$$

2 Trees

In this section we recall the various notions of trees which can be obtained by evaluating λ-terms. As briefly discussed in the introduction, in order to describe trees, it is natural to formalize first the intuitive possible notions of *stable relevant minimal information* coming out of a computation (naturally inducing different notions of *meaningless term* [20]).

If during a computation the following terms appear, their underlined parts will remain stable during the rest (if any) of the computation: $\underline{x}M_1 \ldots M_m$, $\underline{\lambda x.}M$, $P \mathbin{\underline{@}} Q$ (where @ is the explicit representation of the operation of application that is normally omitted, and P is a λ-term which will never reduce to an abstraction). Having a *stable* part in a computation, however, does not necessarily mean that we consider it *relevant*. For instance, we could consider an abstraction $(\underline{\lambda x.}M)$ relevant only in case M is of the form $\lambda y_1 \ldots y_n.zN_1 \ldots N_m$ $(n, m \geq 0)$. This means that we can end up with different notions of *stable relevant minimal information*.

In order to formalize such notions it is possible to define for each notion a reduction relation such that:

(1) if a λ-term can produce stable relevant minimal information, we can get it by means of the given reduction relation;
(2) the computation process represented by the reduction relation stops once stable relevant minimal information is obtained.

In the following we will give a number of reduction relations for λ-terms present in the literature. All are proper restrictions of the usual $\beta\eta$-reduction relation. Syntax, basic notation of the λ-calculus and the usual conventions on variables to avoid explicit α-conversion are as in [4].

A λ-term is a *strong zero term* if it is unsolvable and it cannot be reduced to a lambda abstraction by means of the reduction relation induced by the β-rule [6]. Such terms are called *unsolvables of order* 0 in [22] and *strongly unsolvables* in [1].

Definition 1 *Given the following axioms and rules:*

$$
\begin{array}{lll}
(\beta) & (\lambda x.M)N \to M[N/x] & \\
(\eta) & \lambda x.Mx \to M & \textit{if } x \notin FV(M) \\
(\nu) & M \to N \Rightarrow ML \to NL & \\
(\nu)_{\mathtt{t}} & M \to N \Rightarrow ML \to NL & \text{(provided } M \text{ is not a strong zero term)} \\
(\xi) & M \to N \Rightarrow \lambda x.M \to \lambda x.N &
\end{array}
$$

*we can define the following reduction relations (*RR*) on λ-terms.*

$$
\begin{array}{rll}
\text{(top reduction)} & \to_{\mathtt{t}} & \textit{is the } \mathrm{RR} \textit{ induced by } (\beta) \; and \; (\nu)_{\mathtt{t}} \\
\text{(weak head reduction)} & \to_{\mathtt{w}} & \textit{is the } \mathrm{RR} \textit{ induced by } (\beta) \; and \; (\nu) \\
\text{(head reduction)} & \to_{\mathtt{h}} & \textit{is the } \mathrm{RR} \textit{ induced by } (\beta), (\nu) \; and \; (\xi) \\
\text{(eta head reduction)} & \to_{\mathtt{e}} & \textit{is the } \mathrm{RR} \textit{ induced by } (\beta), (\nu), (\xi) \; and \; (\eta).
\end{array}
$$

The weak head reduction is better known as lazy reduction *[1].*

The sets of λ-terms in normal form with respect to the above defined reduction relations can be described syntactically. Such description makes the different intended notions of stable minimal relevant information explicit.

Definition 2 *(1) A* top normal form [1] *is a term of one of the following three kinds:*
- *(a) an application term of the form $xM_1 \ldots M_m$ $(m \geq 0)$;*
- *(b) an abstraction term $\lambda x.M$;*
- *(c) an application term of the form MN, where M is a strong zero term.*

(2) A weak head normal form *is a term of one of the following two kinds:*
- *(a) an application term of the form $xM_1 \ldots M_m$ $(m \geq 0)$;*
- *(b) an abstraction term $\lambda x.M$.*

(3) A head normal form *is a term of the following kind:*
- *(a) $\lambda x_1 \ldots x_n.yM_1 \ldots M_m$ $(m, n \geq 0)$.*

(4) An eta head normal form *is a term of the following kind:*
- *(a) $\lambda x_1 \ldots x_n.yM_1 \ldots M_m$ $(m, n \geq 0)$, where $x_n \in FV(yM_1 \ldots M_{m-1})$ or $x_n \not\equiv M_m$.*

Notice that the sets of normal forms in the above definition are presented in a proper inclusion order, i.e., the set of top normal forms includes that of weak head normal forms, etc.

Example 3 *(1) For each $n \geq 2$, the term $\Omega_{\mathbf{n}}$ is an example of a strong zero term.*

[1] Called *root stable form* in [19].

(2) $\mathbf{\Omega_2}$ *is not a top normal form, while all* $\mathbf{\Omega_n}$ *(for* $n \geq 3$*) are top normal forms that cannot reduce to weak head normal forms.*
(3) $\lambda x.\mathbf{\Omega_2}$ *is a weak head normal form that cannot reduce to head normal form.*
(4) $\mathbf{\Delta}_\mathbf{2}^\eta$ *is a head normal form but not an eta head normal form.*
(5) $\mathbf{Y}$ *and* $\lambda xy.x(\mathbf{RR}y)$ *are eta head normal forms.*

With this definition we can represent in tree notation all the various related kinds of information we can distract from a λ-term. Given a λ-term M, for each of the four reduction relations we can try to reduce M to normal form. If such a normal form does not exist, then no information is obtainable out of M and its tree is $\perp$. Otherwise, we will put the information thus obtained in a node and build the children of this node by repeating this process on the various subterms of the normal form. In case of head normal forms, this amounts to the usual construction of Böhm trees.

Definition 4 *(1) The* top tree $\mathcal{T}_{\mathtt{t}}(M)$ *of a term* M *is defined by cases as follows:*

- *if* $M \to_{\mathtt{t}} xN_1 \ldots N_m$ $(m \geq 0)$, *then*

$$\mathcal{T}_{\mathtt{t}}(M) = \begin{array}{ccc} & x & \\ \diagup & & \diagdown \\ \mathcal{T}_{\mathtt{t}}(N_1) & \ldots & \mathcal{T}_{\mathtt{t}}(N_m) \end{array}$$

- *if* $M \to_{\mathtt{t}} \lambda x.N$, *then*

$$\mathcal{T}_{\mathtt{t}}(M) = \begin{array}{c} \lambda x \\ | \\ \mathcal{T}_{\mathtt{t}}(N) \end{array}$$

- *if* $M \to_{\mathtt{t}} NP$, *where* N *is a strong zero term, then*

$$\mathcal{T}_{\mathtt{t}}(M) = \begin{array}{ccc} & @ & \\ \diagup & & \diagdown \\ \mathcal{T}_{\mathtt{t}}(N) & & \mathcal{T}_{\mathtt{t}}(P) \end{array}$$

- *otherwise:* $\mathcal{T}_{\mathtt{t}}(M) = \perp$.

(2) The weak tree $\mathcal{T}_{\mathtt{w}}(M)$ *of a term* M *is defined by cases as follows:*

- *if* $M \to_{\mathtt{w}} xN_1 \ldots N_m$ $(m \geq 0)$, *then*

$$\mathcal{T}_{\mathtt{w}}(M) = \begin{array}{ccc} & x & \\ \diagup & & \diagdown \\ \mathcal{T}_{\mathtt{w}}(N_1) & \ldots & \mathcal{T}_{\mathtt{w}}(N_m) \end{array}$$

- *if* $M \to_{\mathtt{w}} \lambda x.N$, *then*

$$\mathcal{T}_{\mathtt{w}}(M) = \begin{array}{c} \lambda x \\ | \\ \mathcal{T}_{\mathtt{w}}(N) \end{array}$$

- *otherwise:* $\mathcal{T}_{\mathtt{w}}(M) = \perp$.

(3) *The* head tree $\mathcal{T}_{\mathrm{h}}(M)$ *of a term* M *is defined by cases as follows:*

- *if* $M \to_{\mathrm{h}} \lambda x_1 \ldots x_n.yN_1 \ldots N_m$ $(n, m \geq 0)$*, then*

$$\mathcal{T}_{\mathrm{h}}(M) = \begin{array}{ccc} & \lambda x_1 \ldots x_n.y & \\ \swarrow & & \searrow \\ \mathcal{T}_{\mathrm{h}}(N_1) & \ldots & \mathcal{T}_{\mathrm{h}}(N_m) \end{array}$$

- *otherwise:* $\mathcal{T}_{\mathrm{h}}(M) = \bot$.

(4) *Let* T *be a head tree, i.e.,* $T \equiv \mathcal{T}_{\mathrm{h}}(M)$*, for some* M*. The* η-normal form *of* T*,* $\eta(T)$*, is defined as follows:*

- $$\eta\left(\begin{array}{ccc} & \lambda x_1 \ldots x_n.y & \\ \swarrow & & \searrow \\ T_1 & \ldots & T_m \end{array}\right) = \begin{cases} \eta\left(\begin{array}{ccc} & \lambda x_1 \ldots x_{n-1}.y & \\ \swarrow & & \searrow \\ T_1 & \ldots & T_{m-1} \end{array}\right) & \textit{proviso } T_m \textit{ is finite, } \eta(T_m) = x_n \neq y \textit{ and } x_n \notin FV(T_i) \textit{ for } 1 \leq i \leq m-1. \\ \begin{array}{ccc} & \lambda x_1 \ldots x_n.y & \\ \swarrow & & \searrow \\ \eta(T_1) & \ldots & \eta(T_m) \end{array} & \textit{otherwise} \end{cases}$$

- $\eta(\bot) = \bot$.

The eta tree $\mathcal{T}_{\mathrm{e}}(M)$ *of a term* M *is defined as* $\eta(\mathcal{T}_{\mathrm{h}}(M))$.

The condition 'T_m is finite' in the above definition is obviously necessary in order to make the latter sound, but it can be easily checked that, in its intended meaning, an η-normal form of a *head* tree can be a variable only when the tree is finite.

One might wonder why the *eta* tree of M is defined through the η-normal form of the *head* tree of M instead of using the eta head normal form of M. As a matter of fact, considering trees instead of terms allows to do more η-reductions, essentially since the set of variables which occur free in $\mathcal{T}_{\mathrm{h}}(M)$ is a *subset* of the set of variables which occur free in M. This was already observed in [4], Remark 10.1.22. Borrowing the example given there, let P be such that $P \to_\beta \lambda z.x(Pz)$, then $\lambda xz.x(Pz)z$ has the reduction behavior and *head* tree as shown in Figure 1. Now, since, as mentioned in [4], there "*z is pushed into infinity*", this tree contains only *one* z, and is therefore an η-redex. This is reflected by the fact that the *eta* tree of the term $\lambda xz.x(Pz)z$ is as in Figure 1.

Finally, the fifth family of trees we shall consider in this paper is the family of the *infinite η-normal forms* of *head* trees (and hence of *eta* trees as well), as defined in [4]. In order to give the definition of *infinite η-normal form*, we need first to recall briefly the definition of *infinite η-expansion* of a variable. Given a variable x, one can consider a (possibly infinite) tree resulting by the limit of a series of expansions like the following:

$$x \;{}_\eta\!\leftarrow \mathcal{T}_{\mathrm{h}}(\lambda y_0.xy_0) \;{}_\eta\!\leftarrow \mathcal{T}_{\mathrm{h}}(\lambda y_0.x(\lambda y_1.y_0y_1)) \;{}_\eta\!\leftarrow \ldots$$

We denote that T is a (possibly infinite) η-expansion of x by $T \geq_\eta x$.

$$
\begin{aligned}
\lambda zx.x(Pz)z &\to_\beta \lambda xz.x((\lambda z.x(Pz))z)z \\
&\to_\beta \lambda xz.x(x(Pz))z \\
&\to_\beta \lambda xz.x(x((\lambda z.x(Pz))z))z \\
&\to_\beta \lambda xz.x(x(x(Pz)))z \\
&\to_\beta \\
&\dots \\
&\to_\beta \lambda xz.x(x(x(x(\cdots))))z
\end{aligned}
$$

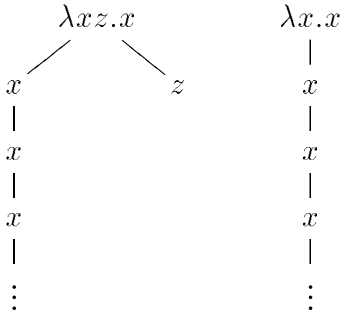

Fig. 1. Reduction path, head tree and eta tree for $\lambda xz.x(Pz)z$, where $P \to_\beta \lambda z.x(Pz)$.

The definition of $\geq_\eta$ requires a formalization of the notion of labeled tree and it is given in the Appendix (Definition 58).

Definition 5 *Let T be a* head *tree, i.e., $T \equiv \mathcal{T}_{\mathtt{h}}(M)$, for some M. The* infinite η-normal form *of T, $\eta_\infty(T)$, is defined as follows:*

- $$\eta_\infty\left(\begin{array}{ccc} & \lambda x_1 \dots x_n.y & \\ \diagup & & \diagdown \\ T_1 & \dots & T_m \end{array}\right) = \begin{cases} \eta_\infty\left(\begin{array}{ccc} & \lambda x_1 \dots x_{n-1}.y & \\ \diagup & & \diagdown \\ T_1 & \dots & T_{m-1} \end{array}\right) & \text{proviso } T_m \geq_\eta x_n,\ x_n \neq y \text{ and } x_n \notin FV(T_i), \text{ for } 1 \leq i \leq m-1 \\ \begin{array}{ccc} & \lambda x_1 \dots x_n.y & \\ \diagup & & \diagdown \\ \eta_\infty(T_1) & \dots & \eta_\infty(T_m) \end{array} & \text{otherwise} \end{cases}$$
- $\eta_\infty(\perp) = \perp$.

The infinite eta tree *$\mathcal{T}_{\mathtt{i}}(M)$ of a term M is defined as $\eta_\infty(\mathcal{T}_{\mathtt{h}}(M))$.*

As mentioned in the introduction, the interest of the tree representations above is that they mimic the local structure (or, equivalently, the λ-theory) of different λ-models.

Example 6 *In Figure 2 we give a few examples of the trees defined above (using the terms of Example 3). They show how trees become less discriminating as we use reduction relations with more rules.*

We will use $\mathcal{T}_\varphi$, with $\varphi \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, to denote the set of trees $\{\mathcal{T}_\varphi(M) \mid M \in \Lambda\}$. Moreover, '$\varphi$-tree' will be short for any tree belonging to $\mathcal{T}_\varphi$. Unless mentioned otherwise, we will assume φ to range over $\{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.

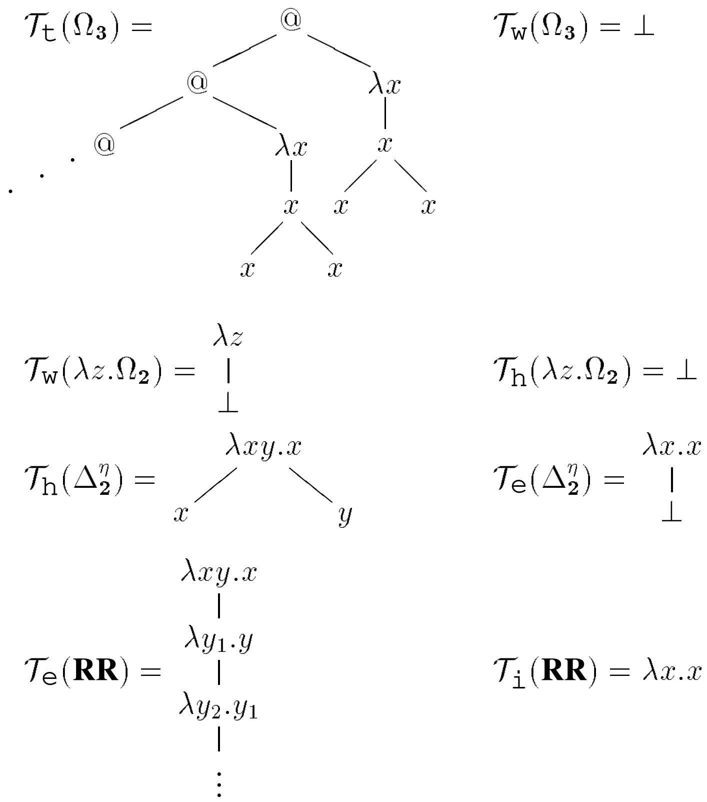

Fig. 2. Trees for Example 6

3 Approximants

Let $\Lambda_\perp$ be the set of terms obtained by adding the symbol $\perp$ to the syntax of the pure λ-calculus. Clearly the tree representations generalize to terms in $\Lambda_\perp$ by assuming $\mathcal{T}_\varphi(\perp) = \perp$. This leaves the set of trees unchanged, i.e., for all $M \in \Lambda_\perp$, there is an $M' \in \Lambda$ such that $\mathcal{T}_\varphi(M) = \mathcal{T}_\varphi(M')$. In fact, M' can be obtained from M by substituting Ω_2 for $\perp$.

It is possible to associate, for any possible notion of stable minimal relevant information, a set of approximants to a λ-term. As usual when dealing with (possibly) infinite structures, one can consider their finite approximations. There are two possible approaches to the definition of approximations of a term M:

- Consider all possible finite trees obtained by pruning the φ-tree of M (the constant $\perp$ is used to represent the (possibly infinite) parts of the trees that have been pruned). Call all these pruned trees φ-approximants of M.
- Consider all possible terms that occur in φ-reduction sequences starting from M (for $\varphi \in \{\mathtt{w}, \mathtt{h}\}$, we should extend the notion of $\to_\varphi$-reduction to $\Lambda_\perp$ by adding

the clause $\perp M \rightarrow \perp$, and also $\lambda x.\perp \rightarrow \perp$ for $\varphi \in \{\mathtt{h}\}$), and calculate their direct approximants (a direct approximant for N is obtained from N by (recursively) replacing (potential) φ-redexes, like $\perp M$ and $\lambda x.\perp$, by $\perp$; to clarify this, one could see this as a generalization of $\beta\perp$-reduction [4]). The φ-approximants of M are now all terms in normal form – with respect to suitable notions of normal form – including $\perp$ that are smaller than those direct approximants.

In the context of β-reduction, these approaches coincide, i.e., for any term M yield the same set.

In the presence of rule (η), both definitions give rise to problems. First of all, in a system with η-reduction, no longer every pruned subtree of the normal form is in normal form itself, a property that holds in a system with just β-reduction. This is caused by the fact that *the number of free occurrences* of a variable will normally decrease by pruning, which affects whether or not a term is an η-redex.

Example 7 *Take the term $\lambda xy.xyy$. The tree below on the left is the* eta *tree of this term, the one on the right is obtained by pruning the first y.*

$$\begin{array}{ccc} & \lambda xy.x & \\ / & & \backslash \\ y & & y \end{array} \qquad\qquad \begin{array}{ccc} & \lambda xy.x & \\ / & & \backslash \\ \perp & & y \end{array}$$

The term in the right-hand tree, $\lambda xy.x\perp y$, is η-reducible.

Also, in the context of η-reduction, the two approaches no longer coincide. For example, take P as defined above. Collecting 'all pruned subtrees' of the *eta* tree of $\lambda xz.x(Pz)z$ yields the set

$$\{\perp, \lambda x.x\perp, \lambda x.x(x\perp), \lambda x.x(x(x\perp)), \ldots\}$$

whereas 'calculate the direct approximants of terms that occur in reduction sequences that start from $\lambda zx.x(Pz)z$' would yield $\{\perp\}$. To understand this, notice that

- none of the reducts of $\lambda xz.x(Pz)z$ is an η-redex, since in all those terms, z appears twice, and
- replacing the redex Pz by $\perp$ in each reduct creates a term that is an η-redex; therefore, for all terms in the sequence, its direct $\beta\eta$-approximant would be $\perp$.

The first set is obviously a better collection of approximants of the infinite tree. Therefore we choose the first approach to define the set of approximants.

Definition 8 *We inductively define the set $\mathcal{A}_\varphi \subseteq \Lambda_\perp$ of approximate normal forms as follows*

(1) $\mathcal{A}_{\mathtt{t}}$ is the smallest subset of $\Lambda_\perp$ such that

(a) if $A_1, \ldots, A_n \in \mathcal{A}_{\mathtt{t}}$, then $xA_1 \ldots A_n \in \mathcal{A}_{\mathtt{t}}$ and $\perp A_1 \ldots A_n \in \mathcal{A}_{\mathtt{t}}$ $(n \geq 0)$,
(b) if $A \in \mathcal{A}_{\mathtt{t}}$, then $\lambda x.A \in \mathcal{A}_{\mathtt{t}}$.

(2) $\mathcal{A}_{\mathtt{w}}$ is the smallest subset of $\Lambda_\perp$ such that
(a) $\perp \in \mathcal{A}_{\mathtt{w}}$,
(b) if $A_1, \ldots, A_n \in \mathcal{A}_{\mathtt{w}}$, then $xA_1 \ldots A_n \in \mathcal{A}_{\mathtt{w}}$ $(n \geq 0)$,
(c) if $A \in \mathcal{A}_{\mathtt{w}}$, then $\lambda x.A \in \mathcal{A}_{\mathtt{w}}$.

(3) $\mathcal{A}_{\mathtt{h}}$ is the smallest subset of $\Lambda_\perp$ such that
(a) $\perp \in \mathcal{A}_{\mathtt{h}}$,
(b) if $A_1, \ldots, A_n \in \mathcal{A}_{\mathtt{h}}$, then $\lambda y_1 \ldots y_m.xA_1 \ldots A_n \in \mathcal{A}_{\mathtt{h}}$ $(m, n \geq 0)$.

(4) $\mathcal{A}_{\mathtt{e}} = \mathcal{A}_{\mathtt{i}} = \mathcal{A}_{\mathtt{h}}$.

We denote the set of approximate normal forms with at most n symbols by $\mathcal{A}_\varphi^{(n)}$.

Let $(M)_\varphi^{(h)}$, where $M \in \Lambda_\perp$, denote the approximate normal form whose φ-tree is the tree obtained out of $\mathcal{T}_\varphi(M)$ by pruning it at height h and inserting the constant $\perp$ as leaves at the end of the cut edges. The formal definition of $(M)_\varphi^{(h)}$ is given in the Appendix (Definition 59).

It is straightforward to verify that $(M)_\varphi^{(h)} \in \mathcal{A}_\varphi$, for all M. For instance, by looking at $\mathcal{T}_{\mathtt{t}}(\Delta_3\Delta_3)$ described above, it is easy to see that $(\Delta_3\Delta_3)_{\mathtt{t}}^{(h)}$, for $h = 0, 1, 2, 3$, are respectively $\perp$, $\perp\perp$, $\perp\perp(\lambda x.\perp)$ and $\perp\perp(\lambda x.\perp)(\lambda x.x\perp\perp)$.

There is a natural partial order between approximants which can be easily formalized by induction.

Definition 9 *The relation $\preceq_\varphi$ is the least partial order on $\mathcal{A}_\varphi$, such that:*

(a) $\perp \preceq_\varphi A$;
(b) if $A \preceq_\varphi A'$, then $\lambda x.A \preceq_\varphi \lambda x.A'$;
(c) if $A \preceq_\varphi A'$ and $B \preceq_\varphi B'$, then $AB \preceq_\varphi A'B'$.

It is easy to verify that $(M)_\varphi^{(h)} \preceq_\varphi (M)_\varphi^{(h+1)}$, for all h. Moreover, pruning trees preserves this order, i.e., if $A \preceq_\varphi B$, then $(A)_\varphi^{(h)} \preceq_\varphi (B)_\varphi^{(h)}$, for all h.

It is possible to associate to a λ-term, for any possible notion of stable minimal relevant information, the set of its approximants, that is the set of all the finite approximations of its corresponding tree.

Definition 10 *The* set $\mathcal{A}_\varphi(M)$ of approximants of $M \in \Lambda$ with respect to the reduction relation φ *is defined by:*

$$\mathcal{A}_\varphi(M) = \{A \in \mathcal{A}_\varphi \mid \exists h. A \preceq_\varphi (M)_\varphi^{(h)}\}.$$

Example 11 • $\mathcal{A}_{\mathtt{t}}(x(\Omega_3\mathbf{I})(\mathbf{II}))$ *contains, for example, approximants like*

$$\bot, x\bot\bot, x(\bot\mathbf{I})\mathbf{I}, x(\bot\Delta_3\mathbf{I})\mathbf{I}, x(\bot\Delta_3\Delta_3\mathbf{I})\mathbf{I}.$$

- $\mathcal{A}_{\mathtt{w}}(\mathbf{I}) = \{\bot, \lambda x.\bot, \mathbf{I}\}$ *while* $\mathcal{A}_{\mathtt{h}}(\mathbf{I}) = \mathcal{A}_{\mathtt{e}}(\mathbf{I}) = \{\bot, \mathbf{I}\}$.
- $\mathcal{A}_{\mathtt{h}}(\lambda xy.xy) = \{\bot, \lambda xy.x\bot, \lambda xy.xy\}$ *while* $\mathcal{A}_{\mathtt{e}}(\lambda xy.xy) = \{\bot, \mathbf{I}\}$.
- *For any term* P *such that* $P \to_\beta \lambda z.x(Pz)$,

$$\mathcal{A}_{\mathtt{e}}(\lambda xz.x(Pz)z) = \{\bot, \lambda x.x\bot, \lambda x.x(x\bot), \dots\}$$

- $\mathcal{A}_{\mathtt{i}}(\mathbf{RR}) = \{\bot, \mathbf{I}\}$, *while both sets* $\mathcal{A}_{\mathtt{h}}(\mathbf{RR})$ *and* $\mathcal{A}_{\mathtt{e}}(\mathbf{RR})$ *are infinite and contain, for instance,* $\bot, \lambda xy.x\bot$, *and* $\lambda xy.x(\lambda y_1.y\bot)$.

Lemma 12 *The* set $\mathcal{A}_\varphi(M)$ *is an ideal, i.e., it is downward closed and directed with respect to* $\preceq_\varphi$.

PROOF. $\mathcal{A}_\varphi(M)$ is downward closed by definition. The fact that $\mathcal{A}_\varphi(M)$ is directed, for all M, follows from the observation that $(M)^{(h)}_\varphi \preceq_\varphi (M)^{(k)}_\varphi$ whenever $h \leq k$. □

Lemma 13 *If* $(M)^{(h)}_\varphi \preceq_\varphi A$ *where* $A \in \mathcal{A}_\varphi(M)$ *then* $(M)^{(h)}_\varphi \equiv (A)^{(h)}_\varphi$.

PROOF. By Definition 10, $A \in \mathcal{A}_\varphi(M)$ implies $A \preceq_\varphi (M)^{(k)}_\varphi$, for some $k \geq h$. By construction, $(M)^{(h)}_\varphi \equiv ((M)^{(k)}_\varphi)^{(h)}_\varphi$. From $(M)^{(h)}_\varphi \preceq_\varphi A \preceq_\varphi (M)^{(k)}_\varphi$ we get $(M)^{(h)}_\varphi \preceq_\varphi (A)^{(h)}_\varphi \preceq_\varphi ((M)^{(k)}_\varphi)^{(h)}_\varphi$, so we conclude $(M)^{(h)}_\varphi \equiv (A)^{(h)}_\varphi$. □

It is natural to expect that our different notions of trees and approximants represent the very same concepts, that is, they formalize the same observational behaviors of λ-terms.

Theorem 14 *For any* M, N:

$$\mathcal{T}_\varphi(M) = \mathcal{T}_\varphi(N) \text{ if and only if } \mathcal{A}_\varphi(M) = \mathcal{A}_\varphi(N).$$

PROOF.

($\Leftarrow$) Reasoning towards a contradiction, we assume that $\mathcal{A}_\varphi(M) = \mathcal{A}_\varphi(N)$ and $(M)^{(h)}_\varphi \not\equiv (N)^{(h)}_\varphi$, for some h. We get $(M)^{(h)}_\varphi \in \mathcal{A}_\varphi(N)$, i.e., by definition $(M)^{(h)}_\varphi \preceq (N)^{(k)}_\varphi$, for some k. We can assume, without loss of generality, that $h \leq k$. Since $(N)^{(k)}_\varphi \in \mathcal{A}_\varphi(M)$, by Lemma 13, we obtain $(M)^{(h)}_\varphi \equiv ((N)^{(k)}_\varphi)^{(h)}_\varphi$. Now $h \leq k$ implies $(N)^{(h)}_\varphi \equiv ((N)^{(k)}_\varphi)^{(h)}_\varphi$ and we are done.

$(\Rightarrow)$ Easy, by definition of $\mathcal{A}_\varphi(\ )$ (Definition 10). $\square$

It is possible to show that $\mathcal{T}_\varphi(M)$ is the least upper-bound of $\mathcal{A}_\varphi(M)$ with respect to $\preceq_\varphi$. We omit the proof of this property here, since it plays no role in this paper.

We extend each partial order $\preceq_\varphi$ to a partial order $\sqsubseteq_\varphi$, which naturally induces an equivalence relation on sets of approximants. This can be proved to coincide with the identity relation on sets of approximants and hence, by Theorem 14, to coincide with the identity on trees.

Definition 15 *(1) The relation $\sqsubseteq_{\mathtt{t}}$ is the least partial order on $\mathcal{A}_{\mathtt{t}}$ that satisfies clauses (a), (b), and (c) of $\preceq_{\mathtt{t}}$ and, moreover:*
(d) $\perp A_1 \dots A_n \sqsubseteq_{\mathtt{t}} x$.

(2) For $\varphi \in \{\mathtt{w}, \mathtt{h}\}$ the relation $\sqsubseteq_\varphi$ is the least partial order on $\mathcal{A}_\varphi$ which satisfies clauses (a), (b), and (c) of $\preceq_\varphi$ and, moreover:
(e) $\lambda y.xA_1 \dots A_n y \sqsubseteq_\varphi xA_1 \dots A_n$, for all variables $y \notin FV(xA_1 \dots A_n)$.

(3) For $\varphi \in \{\mathtt{e}, \mathtt{i}\}$ the relation $\sqsubseteq_\varphi$ is the least partial order on $\mathcal{A}_\varphi$ that satisfies clauses (a), (b), (c), and (e) of $\sqsubseteq_{\mathtt{w}}$ and, moreover:
(f) $xA_1 \dots A_n \sqsubseteq_\varphi \lambda y.xA_1 \dots A_n\perp$, where $x \neq y$.

Note that $A \to_\eta B$ implies $A \sqsubseteq_\varphi B$, for $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$ and $B \sqsubseteq_\varphi A$, for $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. Moreover, we can show that

$$xA_1 \dots A_n \sqsubseteq_\varphi \lambda y_1 \dots y_m.xA_1 \dots A_n \underbrace{\perp \dots \perp}_{m}$$

whenever $x \notin \{y_1, \dots, y_m\}$, for $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. In fact, by clause (f) above, for any $k \geq 0$, we have

$$xA_1 \dots A_n \underbrace{\perp \dots \perp}_{k-1} \sqsubseteq_\varphi \lambda y_k.xA_1 \dots A_n \underbrace{\perp \dots \perp}_{k}.$$

By Definition 9(b) we get

$$\lambda y_1 \dots y_{k-1}.xA_1 \dots A_n \underbrace{\perp \dots \perp}_{k-1} \sqsubseteq_\varphi \lambda y_1 \dots y_k.xA_1 \dots A_n \underbrace{\perp \dots \perp}_{k}.$$

Then, by all such inequalities with $k \leq m$, we are done by transitivity.

It is useful to remark that pruning trees does not preserve these new orders. For instance, $\lambda y.xy \sqsubseteq_\varphi x$, but

$$(\lambda y.xy)^{(1)}_\varphi \equiv \lambda y.x\perp \not\sqsubseteq_\varphi (x)^{(1)}_\varphi \equiv x$$

where $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$. We have a weaker property, namely that if h is the height of the φ-tree of A and $A \sqsubseteq_\varphi B$, then $(A)^{(k)}_\varphi \sqsubseteq_\varphi (B)^{(k)}_\varphi$, for all $k \geq h$.

Definition 16 *For any two terms M and N, we define: $\mathcal{A}_\varphi(M) \simeq_\varphi \mathcal{A}_\varphi(N)$ if and only if, for all $A \in \mathcal{A}_\varphi(M)$, there is $B \in \mathcal{A}_\varphi(N)$ such that $A \sqsubseteq_\varphi B$ and vice versa.*

Lemma 17 *If $A, B \in \mathcal{A}_\varphi(M)$ and $A \sqsubseteq_\varphi B$, then $A \preceq_\varphi B$.*

PROOF. If $A, B \in \mathcal{A}_\varphi(M)$ then $A \preceq_\varphi (M)^{(h)}_\varphi$ and $B \preceq_\varphi (M)^{(k)}_\varphi$, for some $h, k \geq 0$. Let $p = max\{h, k\}$. Then $A \preceq_\varphi (M)^{(p)}_\varphi$ and $B \preceq_\varphi (M)^{(p)}_\varphi$. This means that A and B can be obtained from $(M)^{(p)}_\varphi$ by replacing subterms by $\bot$. Therefore, B cannot be obtained from A either by replacing an occurrence of $\bot A_1 \ldots A_n$ with $n \geq 1$ by x, or by η-reduction, or by replacing an occurrence of $xA_1 \ldots A_n$ by $\lambda y.xA_1 \ldots A_n\bot$. So we can conclude that $A \preceq_\varphi B$. □

Lemma 18 *If $\mathcal{A}_\varphi(M) \simeq_\varphi \mathcal{A}_\varphi(N)$, then $\mathcal{A}_\varphi(M) = \mathcal{A}_\varphi(N)$ and $\mathcal{T}_\varphi(M) = \mathcal{T}_\varphi(N)$.*

PROOF. Reasoning towards a contradiction, assume that for all $A \in \mathcal{A}_\varphi(M)$ there is a $B \in \mathcal{A}_\varphi(N)$ such that $A \sqsubseteq_\varphi B$ (and vice versa). Let $A \in \mathcal{A}_\varphi(M)$ be such that $A \notin \mathcal{A}_\varphi(N)$. Without loss of generality we can assume $A \equiv (M)^{(h)}_\varphi$, for some h. By hypothesis we find $B \in \mathcal{A}_\varphi(N)$ such that $A \sqsubseteq_\varphi B$ and $A' \in \mathcal{A}_\varphi(M)$ such that $B \sqsubseteq_\varphi A'$. We get $A \sqsubseteq_\varphi A'$ which, by Lemma 17, implies $A \preceq_\varphi A'$. Thus, by Lemma 13, $A \equiv (A')^{(h)}_\varphi$. From $A \sqsubseteq_\varphi B \sqsubseteq_\varphi A'$ we get $A \sqsubseteq_\varphi (B)^{(h)}_\varphi \sqsubseteq_\varphi (A')^{(h)}_\varphi$. Hence $A \equiv (B)^{(h)}_\varphi$ and we can conclude $A \in \mathcal{A}_\varphi(N)$. □

The main motivation for the introduction of '$\sqsubseteq_\varphi$' is that it is compatible with the typing that we shall present in the next section.

4 Types and type assignment systems

As stated in the introduction, our static tools to analyze trees (or, equivalently, their corresponding sets of approximants) will be type assignment systems, in particular type assignment systems based on intersection type-like disciplines.

In type assignment systems one derives statements of the form $M : \alpha$, where a term M gets assigned a type α that represents a certain finite information about M. Roughly speaking, a type will be used as a description of a particular notion of normal form. Hence, it is not possible to use a unique set of types to deal with all the trees defined in the previous section. We shall need, instead, three sets of types:

$\mathsf{T}_{\texttt{t}}$ to characterize $\mathcal{T}_{\texttt{t}}$, $\mathsf{T}_{\texttt{wh}}$ to characterize $\mathcal{T}_{\texttt{w}}$ and $\mathcal{T}_{\texttt{h}}$, and $\mathsf{T}_{\texttt{ei}}$ to characterize $\mathcal{T}_{\texttt{e}}$ and $\mathcal{T}_{\texttt{i}}$.

After defining these sets of types, in this section we shall define an order '$\leq_\varphi$' on types that is parameterized by the notion of tree. Then – parametrized by this order – our type assignment systems will be defined (almost) uniformly for all notions of tree. All these type assignment systems deal correctly with terms that carry no information: $\mathcal{T}_\varphi(M) = \perp$ if and only if the universal type ω is the only type that the system related to $\mathcal{T}_\varphi$ can assign to M.

In the following, we shall use the following notation: if $\varphi \in \{\texttt{t}, \texttt{w}, \texttt{h}, \texttt{e}, \texttt{i}\}$, then

$$\bar{\varphi} = \begin{cases} \texttt{t} & \text{if } \varphi = \texttt{t} \\ \texttt{wh} & \text{if } \varphi = \texttt{w} \text{ or } \varphi = \texttt{h} \\ \texttt{ei} & \text{if } \varphi = \texttt{e} \text{ or } \varphi = \texttt{i}. \end{cases}$$

4.1 Types

We start with $\mathsf{T}_{\texttt{t}}$. To describe a top normal form which is the application of two terms, following [7] we will introduce a particular type constructor: the application $\alpha\beta$ of two types α and β. In the intended interpretation a term has type $\alpha\beta$ if its top normal form is the application of two terms, the first one of type α and the second one of type β. We differ from [7] in that we will build types starting only from the unique constant ω, i.e., we won't introduce a new type constant to be interpreted as the set of all strong zero terms.

Some care has to be taken when introducing applicative types, since we have to prevent the presence of inconsistent types. For example, $\omega\omega$ expresses that a top normal form is the application of two terms, the first one being a strong zero term, whereas $\omega\to\omega$ expresses that a top normal form is an abstraction. So we need to prevent their intersection $\omega\omega\wedge(\omega\to\omega)$. Also the type $(\omega\to\omega)\omega$ is meaningless: no top normal form is the application of an abstraction to a term.

We are thus lead to consider a set of 'pretypes' and a smaller set of 'applicative-intersection types', where some obviously inconsistent types, like the ones above, are forbidden. The definition of the set of types is not immediate since, after excluding $\omega\omega\wedge(\omega\to\omega)$ and $(\omega\to\omega)\omega$, we must still decide whether a finite intersection like $(\alpha_1\to\beta_1)\wedge\ldots\wedge(\alpha_n\to\beta_n)$ is empty. The decisive idea comes from Scott's theory of information systems [28]: consistent inputs should give consistent outputs. So, if we interpret the above intersection as the step function which gives an output in $\bigwedge_{i\in I}\beta_i$ whenever the input is in $\bigwedge_{i\in I}\alpha_i$ (where I is a subset of $\{1,\ldots,n\}$), then we must require that if $\bigwedge_{i\in I}\beta_i$ is empty, so is $\bigwedge_{i\in I}\alpha_i$. The definition of types is then obtained by restricting the set of pretypes according to Scott's prescription. This excludes for instance $(\omega\to(\omega\to\omega))\wedge(\omega\to\omega\omega)$, because given an input in ω we would get an output in $(\omega\to\omega)\wedge\omega\omega$, which is impossible since the latter is not a type.

Definition 19 (Pretypes) *The set* PT *of* pretypes *is the set of syntactic expressions inductively defined by:*

(1) $\omega \in \mathsf{PT}$ *(atomic type),*
(2) If $\alpha, \beta \in \mathsf{PT}$*, then* $(\alpha\to\beta)$*,* $(\alpha\beta)$ *and* $(\alpha\wedge\beta)$ *are in* PT.

As usual, in writing types, we assume the following precedence between operators: *application*, *intersection*, *arrow*; we will omit parentheses accordingly. Moreover, we will use $\alpha\to\beta^n\to\gamma$ as short-hand notation for $\alpha\to\underbrace{\beta\to\ldots\to\beta}_{n}\to\gamma$, and $\beta^n\to\gamma$ for $\underbrace{\beta\to\ldots\to\beta}_{n}\to\gamma$.

Definition 20 ($\mathsf{T}_{\mathtt{t}}$) *Given* $\alpha \in \mathsf{PT}$*, we define two predicates '*$\alpha \in \mathsf{T}_{\mathtt{t}}$*' and '*$\alpha \notin \mathsf{T}_{\mathtt{t}}$*' by simultaneous induction on* α*, by stipulating that* $\alpha \in \mathsf{T}_{\mathtt{t}}$ *if and only if one of the following conditions holds (and* $\alpha \notin \mathsf{T}_{\mathtt{t}}$ *if and only if all the conditions do not hold):*

(Universal kind)	α *is* ω.
(Arrow kind)	α *is a finite intersection of the form* $\bigwedge_{i\in I}(\alpha_i\to\beta_i)$*, where* $\alpha_i, \beta_i \in \mathsf{T}_{\mathtt{t}}$ *and, for all* $J \subseteq I$*, either* $\bigwedge_{j\in J}\beta_j \in \mathsf{T}_{\mathtt{t}}$ *or* $\bigwedge_{j\in J}\alpha_j \notin \mathsf{T}_{\mathtt{t}}$.
(Applicative kind)	α *is* $\omega\beta$*, where* $\beta \in \mathsf{T}_{\mathtt{t}}$*, or* α *is a finite intersection of the form* $\bigwedge_{i\in I}\alpha_i\beta_i$*, where* $\bigwedge_{i\in I}\beta_i \in \mathsf{T}_{\mathtt{t}}$*, and* $\bigwedge_{i\in I}\alpha_i \in \mathsf{T}_{\mathtt{t}}$ *is of applicative or universal kind.*

If $\alpha \in \mathsf{T}_{\mathtt{t}}$*, then* $\omega\wedge\alpha \in \mathsf{T}_{\mathtt{t}}$*: the kind of* $\omega\wedge\alpha$ *is defined to be the kind of* α*.*

In what follows, we will consider only types. Also, α, β, γ will range over types of any kind, σ, τ, ρ will range over types of arrow kind (*arrow types*), π, μ, ν will range over types of applicative kind (*applicative types*). Applicative types are only used in the definition of top types.

Without applicative types all the intersections are meaningful. So the definition of $\mathsf{T}_{\mathtt{wh}}$ and $\mathsf{T}_{\mathtt{ei}}$ can be given in a direct way. However, for weak head normal forms and head normal forms, we need to have a new constant, ζ, representing λ-free terms: the constant ω is not enough, as shown by Sangiorgi in [25]. In fact, [25] proves that Δ_2 and $\lambda x.x(\lambda y.xy)$ have the same types when types are built starting from ω using arrow and intersection type constructors. Clearly, these terms have different *weak* and *head* trees. Roughly speaking, ζ can be seen as the collapse of all applicative types.

Definition 21 ($\mathsf{T}_{\mathtt{wh}}$) *The set of types* $\mathsf{T}_{\mathtt{wh}}$ *is inductively defined by*

(1) $\omega, \zeta \in \mathsf{T}_{\mathtt{wh}}$ *(atomic types),*
(2) $\alpha, \beta \in \mathsf{T}_{\mathtt{wh}}$ *imply* $(\alpha \to \beta), (\alpha \wedge \beta) \in \mathsf{T}_{\mathtt{wh}}$.

In order to define $\mathsf{T}_{\mathtt{ei}}$, since terms are considered modulo η, we are forced to equate all atomic types ω, ζ to intersections of arrow types (see [12]). This means that another type constant, ϑ, is needed. In fact, the equations $\zeta = \zeta \to \zeta$ and $\zeta = \omega \to \zeta$ give rise, respectively, to Scott's and Park's D_∞-models as proved in [10]. And the λ-theories of these models are both different from the equality of eta trees.

Definition 22 ($\mathsf{T}_{\mathtt{ei}}$) *The set of types* $\mathsf{T}_{\mathtt{ei}}$ *is inductively defined by*

(1) $\omega, \zeta, \vartheta \in \mathsf{T}_{\mathtt{ei}}$ *(atomic types),*
(2) $\alpha, \beta \in \mathsf{T}_{\mathtt{ei}}$ *imply* $(\alpha \to \beta), (\alpha \wedge \beta) \in \mathsf{T}_{\mathtt{ei}}$.

4.2 Type preorders

On the sets of types of the previous subsection we will define five preorder relations which all take the meaning of ω as universal type, of $\to$ as function space constructor, and of $\wedge$ as intersection into account. The particular properties of these five preorders make them suitable to describe the different trees.

The preorder $\leq_{\mathtt{t}}$, defined on $\mathsf{T}_{\mathtt{t}}$, reflects the interpretation of applicative types. The preorder $\leq_{\mathtt{h}}$, defined on $\mathsf{T}_{\mathtt{wh}}$, equates ω to $\omega \to \omega$, since we want to take the fact that a term like $\lambda x.\bot$ can never be obtained from a head-tree into account. The preorders $\leq_{\mathtt{e}}$ and $\leq_{\mathtt{i}}$ equate all atomic types to arrow types. They differ since, in $\leq_{\mathtt{i}}$, the left-hand subtype of such an arrow type is always ω, while this is not true for $\leq_{\mathtt{e}}$. This difference is essential in order to be able to mimic either infinite or finite η-reductions, as we shall see later.

Definition 23 *(1) We define* $\leq_{\mathtt{t}}$ *as the smallest binary relation over* $\mathsf{T}_{\mathtt{t}}$ *such that:*
(a) it is a preorder in which $\wedge$ *is the meet and* ω *is the top;* [2]
the arrow satisfies:
(b) $\alpha \to \omega \leq \omega \to \omega$*;*
(c) $(\alpha \to \beta) \wedge (\alpha \to \gamma) \leq \alpha \to \beta \wedge \gamma$*;*
(d) $\alpha \geq \alpha'$ *and* $\beta \leq \beta'$ *imply* $\alpha \to \beta \leq \alpha' \to \beta'$;
the applicative types satisfy:
(e) $\pi\alpha \wedge \pi'\alpha' \leq (\pi \wedge \pi')(\alpha \wedge \alpha')$*;*

[2] The explicit axioms and rules are $\alpha \leq \alpha$, $\alpha \leq \beta$ and $\beta \leq \gamma$ imply $\alpha \leq \gamma$, $\alpha \leq \alpha \wedge \alpha$, $\alpha \wedge \beta \leq \alpha$, $\alpha \wedge \beta \leq \beta$, $\alpha \leq \alpha'$ and $\beta \leq \beta'$ imply $\alpha \wedge \beta \leq \alpha' \wedge \beta'$, and $\alpha \leq \omega$.

(f) $\pi \leq \pi'$ *and* $\alpha \leq \alpha'$ *imply* $\pi\alpha \leq \pi'\alpha'$.

(2) We define $\leq_{\mathtt{w}}$ *as the smallest binary relation over* $\mathsf{T}_{\mathtt{wh}}$ *that satisfies the clauses (a) to (d) above.*

(3) We define $\leq_{\mathtt{h}}$ *as the smallest binary relation over* $\mathsf{T}_{\mathtt{wh}}$ *that satisfies the clauses (a), (c) and (d) above and, moreover:*

(g) $\omega \leq \omega\to\omega$.

(4) We define $\leq_{\mathtt{e}}$ *as the smallest binary relation over* $\mathsf{T}_{\mathtt{ei}}$ *that satisfies the clauses (a), (c), (d) and (g) above and, moreover:*

(h) $\zeta \leq \vartheta\to\zeta \leq \zeta$;

(i) $\vartheta \leq \zeta\to\vartheta \leq \vartheta$.

(5) Let $\leq_{\mathtt{i}}$ *be the smallest binary relation over* $\mathsf{T}_{\mathtt{ei}}$ *which satisfies the clauses (a), (c), (d) and (g) above and, moreover:*

(j) $\zeta \leq \omega\to\zeta \leq \zeta$;

(k) $\vartheta \leq \omega\to\vartheta \leq \vartheta$.

'$\alpha =_\varphi \beta$*' is short for '*$\alpha \leq_\varphi \beta$ *and* $\beta \leq_\varphi \alpha$*'.*

Notice that clause (b) is derivable from clause (g), so it is safe to eliminate clause (b) from the definitions of $\leq_{\mathtt{h}}$, $\leq_{\mathtt{e}}$, and $\leq_{\mathtt{i}}$.

Example 24 • $\omega\wedge\alpha =_\varphi \alpha$, *for every type* $\alpha \in \mathsf{T}_{\bar\varphi}$.
• $(\omega\to\omega)\wedge\sigma =_{\mathtt{t}} \sigma$, *for every arrow type* $\sigma \in \mathsf{T}_{\mathtt{t}}$.
• $(\omega\to\omega)\wedge\alpha =_\varphi \alpha$, *for every type* $\alpha \in \mathsf{T}_{\bar\varphi}$ *(but for the case* $\alpha =_{\mathtt{w}} \omega$ *when* $\varphi = \mathtt{w}$*) where* $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.
• $\omega\omega\wedge\pi =_{\mathtt{t}} \pi$, *for every applicative type* $\pi \in \mathsf{T}_{\mathtt{t}}$.

We need to consider some properties of $\leq_{\mathtt{t}}$ already proved in [7] for the sets of types and the preorder relations there introduced.

Lemma 25 *(1) If* $\bigwedge_{i\in I}(\alpha_i\to\beta_i) \leq_{\mathtt{t}} \gamma <_{\mathtt{t}} \omega$, *or* $\gamma \leq_{\mathtt{t}} \bigwedge_{i\in I}(\alpha_i\to\beta_i)$, *then* γ *is an arrow type.*

(2) If $\pi \leq_{\mathtt{t}} \alpha <_{\mathtt{t}} \omega$, *or* $\alpha \leq_{\mathtt{t}} \pi$, *then* α *is an applicative type.*

(3) $\pi\alpha \leq_{\mathtt{t}} \pi'\alpha'$ *implies* $\pi \leq_{\mathtt{t}} \pi'$ *and* $\alpha \leq_{\mathtt{t}} \alpha'$.

(4) $\pi\alpha\wedge\pi'\alpha' =_{\mathtt{t}} (\pi\wedge\pi')(\alpha\wedge\alpha')$.

(5) For any applicative type π, $\pi =_{\mathtt{t}} \omega\alpha_1 \ldots \alpha_n$, *for some* $n, \alpha_1, \ldots, \alpha_n$.

(6) Assume $\beta \leq_{\mathtt{t}} \alpha_1$ *and* $\beta \leq_{\mathtt{t}} \alpha_2$. *If* $\beta \in \mathsf{T}_{\mathtt{t}}$, *then* $\alpha_1\wedge\alpha_2 \in \mathsf{T}_{\mathtt{t}}$.

PROOF.

(1) – (3) By induction on the definition of $\leq_{\mathtt{t}}$.

(4) In fact, $\pi\alpha\wedge\pi'\alpha' \leq_{\mathtt{t}} (\pi\wedge\pi')(\alpha\wedge\alpha')$ follows from clause (e) of Definition 23. The converse follows from clause (f) of the same definition and the fact that $\beta \leq_{\mathtt{t}} \gamma$ and $\beta \leq_{\mathtt{t}} \delta$ imply $\beta \leq_{\mathtt{t}} \gamma\wedge\delta$.

(5) First observe that $\omega\wedge\pi =_{\mathtt{t}} \pi$ for all types π. Then, by (4), we are done.

(6) By cases, using (1) – (3). ☐

All the pre-orders we introduced enjoy the following two properties which can be shown by induction on $\leq_\varphi$. The first property says that an arrow type terminating with an atom is $\wedge$-prime[3]. The second essentially says that the sets of types that are filters represent the space of continuous functions (see [10]).

Lemma 26 *(1) If $\alpha\wedge\beta \leq_\varphi \gamma_1\to\ldots\to\gamma_n\to\delta$, where δ is atomic and $n \geq 0$, then either $\alpha \leq_\varphi \gamma_1\to\ldots\to\gamma_n\to\delta$ or $\beta \leq_\varphi \gamma_1\to\ldots\to\gamma_n\to\delta$.*
(2) If $\bigwedge_{i\in I}(\alpha_i\to\beta_i) \leq_\varphi \alpha\to\beta$, where $\beta \neq_\varphi \omega$, then for some $J \subseteq I$ we get $\alpha \leq_\varphi \bigwedge_{j\in J}\alpha_j$ and $\bigwedge_{j\in J}\beta_j \leq_\varphi \beta$.

As an immediate consequence of Lemma 26(2) we get that $\alpha\to\beta \leq_\varphi \gamma\to\delta$ implies $\gamma \leq_\varphi \alpha$ and $\beta \leq_\varphi \delta$.

4.3 Type assignment systems

For each preorder introduced in the previous subsection, we will define a type assignment system associating λ-terms to types belonging to the domain of the preorder. As said at the beginning of this section, these systems can be defined *almost* uniformly. In fact, there are six rules which are common to all systems and which are standard in intersection type disciplines. The type assignment systems $\vdash_\varphi$ ($\varphi \in \{\mathtt{w},\mathtt{h},\mathtt{e},\mathtt{i}\}$) are defined by six such rules, and instantiating rule ($\leq_\varphi$) with the corresponding preorder. However, to define $\vdash_{\mathtt{t}}$ we have to deal with applicative types, and hence we need two extra rules: (ω*app*) and (*app*).

These rules for applicative types allow to deduce the type $\alpha\beta$ for the application MN when M has type α, N has type β and M is a strong zero term. Moreover, a rule ($\beta-$*exp*) is needed as well, since applicative types are not invariant under β-expansion of subjects. For example, without ($\beta-$*exp*) we derive $\vdash_{\mathtt{t}} \Omega_2\mathbf{I} : \omega(\omega\to\omega)$, but we cannot derive $\vdash_{\mathtt{t}} (\lambda xy.y\Delta_2 x)\mathbf{I}\Delta_2 : \omega(\omega\to\omega)$.

A *basis* Γ is a (finite or infinite) set of statements of the shape $x{:}\alpha$, with distinct variables as subjects. In writing $\Gamma, x{:}\alpha$ we assume that x does not occur in Γ. We denote by $\mathcal{B}_{\mathtt{t}}$, $\mathcal{B}_{\mathtt{wh}}$, $\mathcal{B}_{\mathtt{ei}}$ the sets of bases whose predicates belong to $\mathsf{T}_{\mathtt{t}}$, $\mathsf{T}_{\mathtt{wh}}$, and $\mathsf{T}_{\mathtt{ei}}$, respectively.

Definition 27 (Type assignment systems) *Consider the rules of Figure 3:*

(1) The type assignment system $\vdash_{\mathtt{t}}$ is defined by the rules (Ax), (ω), ($\to$I), ($\to$E), ($\wedge$I), (ωapp), (app), ($\beta-$exp), and ($\leq_{\mathtt{t}}$), where $\Gamma \in \mathcal{B}_{\mathtt{t}}$, $\pi \in \mathsf{T}_{\mathtt{t}}$ is an applicative type, and $\alpha,\beta \in \mathsf{T}_{\mathtt{t}}$.
(2) The type assignment system $\vdash_\varphi$, for $\varphi \in \{\mathtt{w},\mathtt{h},\mathtt{e},\mathtt{i}\}$ is defined by the rules (Ax), (ω), ($\to$I), ($\to$E), ($\wedge$I), and ($\leq_\varphi$), where $\Gamma \in \mathcal{B}_{\bar\varphi}$ and $\alpha,\beta \in \mathsf{T}_{\bar\varphi}$.

[3] A type γ is called $\wedge$-prime, if and only if $\alpha\wedge\beta \leq \gamma$ implies $\alpha \leq \gamma$ or $\beta \leq \gamma$.

$$(Ax)\ \frac{}{\Gamma, x{:}\alpha \vdash x : \alpha} \qquad (\omega)\ \frac{}{\Gamma \vdash M : \omega}$$

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

$$(\wedge I)\ \frac{\Gamma \vdash M : \alpha \qquad \Gamma \vdash M : \beta}{\Gamma \vdash M : \alpha\wedge\beta} \qquad (\leq_\varphi)\ \frac{\Gamma \vdash M : \alpha \qquad \alpha \leq_\varphi \beta}{\Gamma \vdash M : \beta}$$

$$(\beta{-}exp)\ \frac{\Gamma \vdash N : \alpha \qquad M \to_\beta N}{\Gamma \vdash M : \alpha} \qquad (app)\ \frac{\Gamma \vdash M : \pi \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \pi\alpha}$$

$$(\omega app)\ \frac{M \text{ is a strong zero term} \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \omega\alpha}$$

Note: M, N are terms of $\Lambda_\perp$, and, in $(\beta{-}exp)$, the relation $\to_\beta$ is the *full* β-reduction, i.e., the symmetric, transitive and compatible closure of rule (β).

Fig. 3. Derivation rules

Example 28 • $\vdash_{\mathtt{t}}\ \Omega_{\mathbf{3}} : \omega(\omega\to\omega)$, *whereas, in all other systems, any type deducible for* $\Omega_{\mathbf{3}}$ *is equivalent to* ω.

- $\vdash_\varphi\ \lambda x.\Omega_{\mathbf{2}} : \omega\to\omega$, *for* $\varphi \in \{\mathtt{t}, \mathtt{w}\}$, *whereas, in all other systems, any type deducible for* $\lambda x.\Omega_{\mathbf{2}}$ *is equivalent to* ω.
- $\vdash_\varphi \Delta_{\mathbf{2}} : \zeta\wedge(\zeta\to\zeta)\to\zeta$ *for* $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, *and also* $\vdash_\varphi \Delta_{\mathbf{2}}{}^\eta : \zeta\wedge(\zeta\to\zeta)\to\zeta$, *for* $\varphi \in \{\mathtt{e}, \mathtt{i}\}$, *whereas the latter statement is not deducible for* $\varphi \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}\}$.
- $\vdash_\varphi \mathbf{I} : \zeta\to\zeta$, *for* $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, *and also* $\vdash_{\mathtt{i}}$ **RR** $: \zeta\to\zeta$, *whereas the latter statement is not deducible in all other systems.*

Remark 29 *At a first glance one could wonder whether, in the definition of* $\vdash_{\mathtt{t}}$, *it is possible to eliminate rule* $(\beta{-}exp)$ *by replacing rule* (ωapp) *by a rule like*

$$(\omega app')\ \frac{M \to_\beta ZN \qquad Z \text{ is a strong zero term} \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash M : \omega\alpha}$$

This is not the case. In fact, it is easy to check that $(\lambda y.x(y\Delta_{\mathbf{2}}\mathbf{I}))\Delta_{\mathbf{2}} \to_\beta x(\Omega_{\mathbf{2}}\mathbf{I})$ *and* $x{:}\omega\omega\to\alpha \vdash_{\mathtt{t}} x(\Omega_{\mathbf{2}}\mathbf{I}) : \alpha$. *However, without rule* $(\beta{-}exp)$, *we cannot derive* $x{:}\omega\omega\to\alpha \vdash_{\mathtt{t}} (\lambda y.x(y\Delta_{\mathbf{2}}\mathbf{I}))\Delta_{\mathbf{2}} : \alpha$.

Since terms are considered modulo α-conversion, the weakening rule is admissible. Moreover, as usual, we have $\Gamma_{|M} \vdash_\varphi M : \alpha$ whenever $\Gamma \vdash_\varphi M : \alpha$, where $\Gamma_{|M} = \{x{:}\beta \in \Gamma \mid x \in FV(M)\}$ [4].

We define *Dom*$(\Gamma) = \{x \mid x{:}\alpha \in \Gamma$ and $\alpha \neq \omega\}$ and we assume $x{:}\omega \in \Gamma$ whenever

[4] For rule $(\beta{-}exp)$ note that $M \to_\beta N$ implies $FV(N) \subseteq FV(M)$.

$x \notin Dom(\Gamma)$. This is sound in view of rule (ω).

We want to consider unions of bases taking the intersections of the types with the same subjects. Since not all intersections of types in $\mathsf{T}_{\mathtt{t}}$ are types, we need to allow in this case only unions of compatible bases, according to the following definition. For the other sets of types, any two arbitrary bases are compatible.

Definition 30 *We say that two bases $\Gamma, \Gamma' \in \mathcal{B}_{\bar{\varphi}}$ are* compatible *if and only if $x{:}\alpha \in \Gamma$ and $x{:}\beta \in \Gamma'$ imply $\alpha\wedge\beta \in \mathsf{T}_{\bar{\varphi}}$. If Γ and Γ' are compatible, we define their union $\uplus$ as*

$$\Gamma \uplus \Gamma' = \{x{:}\alpha\wedge\beta \mid x{:}\alpha \in \Gamma \textit{ and } x{:}\beta \in \Gamma'\}.$$

Notice that $x{:}\alpha\wedge\omega \in \Gamma \uplus \Gamma'$ whenever $x{:}\alpha \in \Gamma$ and $x \notin Dom(\Gamma')$, since by convention we get $x{:}\omega \in \Gamma'$. Similarly, when $x{:}\beta \in \Gamma'$ and $x \notin Dom(\Gamma)$.

As expected, we have generation lemmas for all the given type assignment systems. To avoid the use of rule $(\beta{-}exp)$, the generation lemma for $\vdash_{\mathtt{t}}$ considers approximate normal forms instead of arbitrary terms.

Lemma 31 (Generation Lemma for $\vdash_{\mathtt{t}}$) *Let $A \in \mathcal{A}_{\mathtt{t}}$.*

(1) $\Gamma \vdash_{\mathtt{t}} \bot : \alpha$ *implies* $\alpha =_{\mathtt{t}} \omega$;
(2) *If* $\Gamma \vdash_{\mathtt{t}} A : \alpha$, $\alpha \neq_{\mathtt{t}} \omega$, *and*
- *(a)* $A \equiv x$, *then* $x{:}\beta \in \Gamma$ *for some* $\beta \leq_{\mathtt{t}} \alpha$;
- *(b)* $A \equiv \lambda x.A'$, *then* $\alpha =_{\mathtt{t}} \bigwedge_{i\in I}(\alpha_i{\to}\beta_i)$ *and, for* $i \in I$, $\Gamma, x{:}\alpha_i \vdash_{\mathtt{t}} A' : \beta_i$;
- *(c)* $A \equiv xA_1 \ldots A_nA'$, *then there exists* β *such that* $\Gamma \vdash_{\mathtt{t}} A' : \beta$, *and either* $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta{\to}\alpha$, *or* $\alpha \geq_{\mathtt{t}} \pi\beta$ *and* $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi$, *for some* π;
- *(d)* $A \equiv \bot A_1 \ldots A_nA'$, *then there is* β *such that* $\Gamma \vdash_{\mathtt{t}} A' : \beta$, $\alpha \geq_{\mathtt{t}} \pi\beta$ *and* $\Gamma \vdash_{\mathtt{t}} \bot A_1 \ldots A_n : \pi$, *for some* π;

(3) *If* $\Gamma \vdash_{\mathtt{t}} A : \alpha$ *and* $\Gamma \vdash_{\mathtt{t}} A : \beta$, *then* $\alpha\wedge\beta \in \mathsf{T}_{\mathtt{t}}$.

PROOF. The proof for (1) is immediate. We prove (2) and (3) by simultaneous induction on A, showing each time first (2) by a secondary induction on derivations.

- $A \equiv x$.
  (2a) Follows easily by induction on derivations, using the transitivity of $\leq_{\mathtt{t}}$.
  (3) Follows from (2a) and Lemma 25(6).
- $A \equiv \lambda x.A'$.
  (2b) Proved by induction on derivations. If the last applied rule is $(\leq_{\mathtt{t}})$, $\alpha =_{\mathtt{t}} \bigwedge_{i\in I}(\alpha_i{\to}\beta_i)$ follows from Lemma 25(1) and $\Gamma, x{:}\alpha_i \vdash_{\mathtt{t}} A' : \beta_i$ follows from Lemma 26(2).
  (3) Let, by (2b), $\alpha =_{\mathtt{t}} \bigwedge_{i\in I}(\alpha_i{\to}\beta_i)$, $\beta =_{\mathtt{t}} \bigwedge_{j\in J}(\alpha_j{\to}\beta_j)$. Consider $K \subseteq I \cup J$: if $\bigwedge_{k\in K}\alpha_k \notin \mathsf{T}_{\mathtt{t}}$ there is no problem. If $\bigwedge_{k\in K}\alpha_k \in \mathsf{T}_{\mathtt{t}}$, we have by

(2b) that $\Gamma, x{:}\alpha_k \vdash_{\mathtt{t}} A' : \beta_k$, for all $k \in K$, therefore using rule $\leq_{\mathtt{t}}$ we have $\Gamma, x{:} \bigwedge_{k\in K} \alpha_k \vdash_{\mathtt{t}} A' : \beta_k$, for all $k \in K$. This implies, by induction, $\bigwedge_{k\in K} \beta_k \in \mathsf{T}_{\mathtt{t}}$, so we conclude $\alpha\wedge\beta \in \mathsf{T}_{\mathtt{t}}$.

- $A \equiv xA_1 \ldots A_n A'$.

  (2c) By induction on derivations. The only interesting case is when the last applied rule is $(\wedge I)$

$$(\wedge I)\ \frac{\Gamma \vdash_{\mathtt{t}} A : \alpha_1 \qquad \Gamma \vdash_{\mathtt{t}} A : \alpha_2}{\Gamma \vdash_{\mathtt{t}} A : \alpha_1\wedge\alpha_2}.$$

  By the second induction, there are β_i ($i = 1, 2$), such that $\Gamma \vdash_{\mathtt{t}} A' : \beta_i$, and either $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta_i \to \alpha_i$, or $\alpha_i \geq_{\mathtt{t}} \pi_i\beta_i$ and $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi_i$, for some π_i. By induction on (3), we cannot have $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta_1\to\alpha_1$ and $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi_2$, or $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta_2\to\alpha_2$ and $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi_1$. Moreover, we get $\beta_1\wedge\beta_2 \in \mathsf{T}_{\mathtt{t}}$ and either $(\beta_1\to\alpha_1)\wedge(\beta_2\to\alpha_2) \in \mathsf{T}_{\mathtt{t}}$ or $\pi_1\wedge\pi_2 \in \mathsf{T}_{\mathtt{t}}$. Therefore, using rules $(\leq_{\mathtt{t}})$ and $(\wedge I)$, $\Gamma \vdash_{\mathtt{t}} A' : \beta_1\wedge\beta_2$ and either $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta_1\wedge\beta_2\to\alpha_1\wedge\alpha_2$ or $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi_1\wedge\pi_2$.

  (3) Let $\Gamma \vdash_{\mathtt{t}} A : \alpha_1$ and $\Gamma \vdash_{\mathtt{t}} A : \alpha_2$. By induction on (2c) there are β_i, for $i = 1, 2$ such that $\Gamma \vdash_{\mathtt{t}} A' : \beta_i$, and either $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \beta_i\to\alpha_i$, or $\Gamma \vdash_{\mathtt{t}} xA_1 \ldots A_n : \pi_i$ and $\alpha_i \geq_{\mathtt{t}} \pi_i\beta_i$,, for some π_i. So we can conclude as above that $\beta_1\wedge\beta_2 \in \mathsf{T}_{\mathtt{t}}$ and either $(\beta_1\to\alpha_1)\wedge(\beta_2\to\alpha_2) \in \mathsf{T}_{\mathtt{t}}$ or $\pi_1\wedge\pi_2 \in \mathsf{T}_{\mathtt{t}}$. This implies either $\alpha_1\wedge\alpha_2 \in \mathsf{T}_{\mathtt{t}}$ or $(\pi_1\wedge\pi_2)(\beta_1\wedge\beta_2) \in \mathsf{T}_{\mathtt{t}}$. In the second case we get $(\pi_1\wedge\pi_2)(\beta_1\wedge\beta_2) \leq_{\mathtt{t}} \pi_1\beta_1\wedge\pi_2\beta_2$ by Lemma 25(4). This implies $(\pi_1\wedge\pi_2)(\beta_1\wedge\beta_2) \leq_{\mathtt{t}} \alpha_1\wedge\alpha_2$, so we can conclude $\alpha_1\wedge\alpha_2 \in \mathsf{T}_{\mathtt{t}}$ by Lemma 25(6).

- $A \equiv \bot A_1 \ldots A_n A'$. The proof of this case is similar to and simpler than that of the previous case. □

The set of types deducible in $\vdash_\varphi$ for approximate normal forms is not decreasing with respect to the order relation $\preceq_\varphi$ between approximate normal forms. From this we easily obtain a consistency property between the types deducible for the approximants of the same term in $\vdash_{\mathtt{t}}$.

Lemma 32 *(1) If $\Gamma \vdash_\varphi A : \alpha$ and $A \preceq_\varphi A'$, then $\Gamma \vdash_\varphi A' : \alpha$.*

(2) If $A, A' \in \mathcal{A}_{\mathtt{t}}(M)$, then a basis $\Gamma \in \mathcal{B}_{\mathtt{t}}$ cannot assign an arrow type to A and an applicative type to A'.

PROOF.

(1) By induction on the definition of $\preceq_\varphi$.

(2) Since '$\preceq_{\mathtt{t}}$' is directed (Lemma 12), reasoning towards a contradiction we would get a single approximate normal form which has both an arrow and an applicative type. This is impossible by Lemma 31(3) because the intersection of an applicative type and an arrow type is not a type. □

Lemma 33 (Generation Lemma for $\vdash_\varphi$) *Let $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.*

(1) $\Gamma \vdash_\varphi \bot : \alpha$ implies $\alpha =_\varphi \omega$;
(2) $\Gamma \vdash_\varphi x : \alpha$ if and only if $x{:}\beta \in \Gamma$, for some $\beta \leq_\varphi \alpha$;
(3) $\Gamma \vdash_\varphi \lambda x.M : \alpha$ (and $\alpha \neq_{\mathtt{w}} \omega$ when $\varphi = \mathtt{w}$) if and only if $\alpha =_\varphi \bigwedge_{i\in I}(\alpha_i \to \beta_i)$ and, for $i \in I$, $\Gamma, x{:}\alpha_i \vdash_\varphi M : \beta_i$;
(4) $\Gamma \vdash_\varphi MN : \alpha$ if and only if there is β such that $\Gamma \vdash_\varphi M : \beta \to \alpha$, and $\Gamma \vdash_\varphi N : \beta$.

PROOF. All points can be shown by standard induction on the structure of derivations, using Lemma 26(2) for (3). □

With a standard proof we can show that rule $(\beta{-}exp)$ is admissible in the systems $\vdash_\varphi$, for $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$. Moreover, types are preserved by η expansion in $\vdash_{\mathtt{e}}$ and $\vdash_{\mathtt{i}}$.

Theorem 34

(1) Let $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$. Then $\Gamma \vdash_\varphi M[N/x] : \alpha$ implies $\Gamma \vdash_\varphi (\lambda x.M)N : \alpha$.
(2) Let $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. Then $\Gamma \vdash_\varphi M : \alpha$ and $x \notin FV(M)$ imply $\Gamma \vdash_\varphi \lambda x.Mx : \alpha$.
(3) Let $\varphi \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$. Then $\Gamma \vdash_\varphi N : \alpha$ and $M \to_\beta N$ imply $\Gamma \vdash_\varphi M : \alpha$.
(4) Let $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. Then $\Gamma \vdash_\varphi N : \alpha$ and $M \to_\eta N$ imply $\Gamma \vdash_\varphi M : \alpha$.

PROOF.

(1) Let $\Gamma_i \vdash_\varphi N : \beta_i$, for $1 \leq i \leq n$ and $n \geq 0$, be all the statements whose subject is N in a derivation of $\Gamma \vdash_\varphi M[N/x] : \alpha$. Notice that $\Gamma \subseteq \Gamma_i$ but $\Gamma_{|N} = \Gamma_{i|N}$, for all $1 \leq i \leq n$. So we can derive $\Gamma \vdash_\varphi N : \bigwedge_{1\leq i\leq n}\beta_i$, with the convention that $\bigwedge_{1\leq i\leq n}\beta_i = \omega$ whenever $n = 0$. One can easily see, by induction on M, that $\Gamma, x{:}\bigwedge_{1\leq i\leq n}\beta_i \vdash_\varphi M : \alpha$. Then, by rule $(\to I)$, we get $\Gamma \vdash_\varphi \lambda x.M : \bigwedge_{1\leq i\leq n}\beta_i \to \alpha$. Hence, by rule $(\to I)$, we can conclude $\Gamma \vdash_\varphi (\lambda x.M)N : \alpha$.
(2) By easy induction on α, taking into account that each atomic type is equal to an arrow type in the preorders $\leq_{\mathtt{e}}$ and $\leq_{\mathtt{i}}$.
(3) – (4) By straightforward induction, using, respectively, (1) and (2). □

For the type assignment system $\vdash_{\mathtt{i}}$ we need a further property dealing with the types we can deduce for the terms whose infinite eta tree is just one variable. The notion of strict types comes in handy [2].

Definition 35 *The set of* strict types $\mathsf{ST} \subseteq \mathsf{T}_{\mathtt{ei}}$ *is the minimal set such that:*

(1) $\omega, \zeta, \vartheta \in \mathsf{ST}$,
(2) $\alpha, \beta_1, \ldots, \beta_n \in \mathsf{ST},\ n \geq 1 \ \Rightarrow\ \beta_1 \wedge \ldots \wedge \beta_n \to \alpha \in \mathsf{ST}$.

Proposition 36 *For all types $\alpha \in \mathsf{T}_{\mathtt{ei}}$, there is a set of strict types $\beta_1, \ldots, \beta_n \in \mathsf{ST}$ such that $\alpha =_{\mathtt{i}} \beta_1 \wedge \ldots \wedge \beta_n$.*

PROOF. By induction on α. Observe that $\gamma \to \delta_1 \wedge \delta_2 =_{\mathtt{i}} (\gamma \to \delta_1) \wedge (\gamma \to \delta_2)$. □

We will now introduce a measure on types which gives us, for each equivalence class, the number of symbols occurring in the 'minimal' intersection of strict types.

Definition 37 *(1) Define $|\ | : \mathsf{T}_{\mathtt{ei}} \to \mathbb{N}$ by:*
(a) $|\omega| = |\zeta| = |\vartheta| = 1$,
(b) $|\alpha \to \beta| = |\alpha \wedge \beta| = |\alpha| + |\beta| + 1$.
(2) Define $||\ || : \mathsf{T}_{\mathtt{ei}} \to \mathbb{N}$ by $||\alpha|| = min\{|\beta_1 \wedge \ldots \wedge \beta_n| \mid \beta_i \in \mathsf{ST}$ for $1 \leq i \leq n$ and $\beta_1 \wedge \ldots \wedge \beta_n =_{\mathtt{i}} \alpha\}$.

Theorem 38 *Let $\mathcal{T}_{\mathtt{i}}(M) \geq_\eta x$.*

(1) $x{:}\alpha \vdash_{\mathtt{i}} M : \alpha$.
(2) If $\Gamma \vdash_{\mathtt{i}} x : \alpha$ then $\Gamma \vdash_{\mathtt{i}} M : \alpha$.

PROOF.

(1) By induction on $||\alpha||$. Notice that $\mathcal{T}_{\mathtt{i}}(M) \geq_\eta x$ implies

$$M =_\beta \lambda y_1 \ldots y_n . x M_1 \ldots M_n$$

where $\mathcal{T}_{\mathtt{i}}(M_i) \geq_\eta y_i$, for $1 \leq i \leq n$ and $n \geq 0$.

- If $||\alpha|| = 1$ then $\alpha =_{\mathtt{i}} \omega$, $\alpha =_{\mathtt{i}} \zeta$ or $\alpha =_{\mathtt{i}} \vartheta$. The case $\alpha =_{\mathtt{i}} \omega$ is trivial. Otherwise, we derive $\vdash_{\mathtt{i}} M_i : \omega$, for $i \leq i \leq n$, by rule (ω) and we conclude $x{:}\xi \vdash_{\mathtt{i}} M : \xi$, for $\xi \in \{\zeta, \vartheta\}$, using rules $(\leq_{\mathtt{i}})$, $(\to E)$, and $(\to I)$, since $\zeta =_{\mathtt{i}} \omega^n \to \zeta$ and $\vartheta =_{\mathtt{i}} \omega^n \to \vartheta$.
- For the induction step, by Proposition 36, we can assume, without loss of generality, that α is a strict type. We distinguish two subcases.
  - $\alpha =_{\mathtt{i}} \beta_1 \to \ldots \to \beta_n \to \gamma$ with $||\alpha|| = ||\beta_1|| + \ldots + ||\beta_n|| + ||\gamma|| + n$ By induction $y_i{:}\beta_i \vdash_{\mathtt{i}} M_i : \beta_i$, for $1 \leq i \leq n$, and so we obtain $x{:}\alpha \vdash_{\mathtt{i}} M : \alpha$, using rules $(\leq_{\mathtt{i}})$, $(\to E)$, and $(\to I)$.
  - $\alpha =_{\mathtt{i}} \beta_1 \to \ldots \to \beta_m \to \xi$ with $m < n$, $\xi \in \{\zeta, \vartheta\}$ and $||\alpha|| = ||\beta_1|| + \ldots + ||\beta_m|| + m + 1$. Also, by induction, $y_i{:}\beta_i \vdash_{\mathtt{i}} M_i : \beta_i$, for $1 \leq i \leq m$. Moreover, by rule (ω), we get $\vdash_{\mathtt{i}} M_i : \omega$, for $m+1 \leq i \leq n$. We conclude as in previous case, since $\alpha =_{\mathtt{i}} \beta_1 \to \ldots \to \beta_m \to \omega^{n-m} \to \xi$.

(2) Follows easily from (1) and Lemma 33(2). □

Our type assignment systems enjoy the approximation property, i.e., we can deduce a type for a term M if and only if we can deduce this type for an approximant of M, with respect to the relative notion of approximant (Theorem 43). Such a theorem, interesting in its own right, will be used in the next section to show that our type assignment systems are tools to analyze the observational behavior represented by trees.

We prove the Approximation Theorem by means of a variant of Tait's 'computability' technique. We define sets of 'approximable' and 'computable' terms. The computable terms are defined by induction on types (Definition 39), and every computable term is shown to be approximable (Lemma 41(2)). Using induction on type derivations, we show that every term is computable for the appropriate type (Lemma 42).

It is useful to have a short-hand notation for the property we want to show. We define '$\mathsf{App}_{\varphi}(\Gamma, \alpha, M)$' as an abbreviation for '$\exists A \in \mathcal{A}_{\varphi}(M).\Gamma \vdash_{\varphi} A : \alpha$'.

Definition 39 *We define the predicate* $\mathsf{Comp}_{\varphi}(\Gamma, \alpha, M)$ *by induction on* $\alpha \in \mathsf{T}_{\bar{\varphi}}$ *as follows:*

(1) $\mathsf{Comp}_{\varphi}(\Gamma, \omega, M)$ *is always true;*
(2) $\mathsf{Comp}_{\mathtt{t}}(\Gamma, \pi, M)$*, if and only if* $\mathsf{App}_{\mathtt{t}}(\Gamma, \pi, M)$*, for every type* π *of applicative kind;*
(3) $\mathsf{Comp}_{\varphi}(\Gamma, \zeta, M)$*, if and only if* $\mathsf{App}_{\varphi}(\Gamma, \zeta, M)$*, for* $\varphi \in \{\mathtt{w}, \mathtt{h}\}$*;*
(4) $\mathsf{Comp}_{\mathtt{e}}(\Gamma, \zeta, M)$*, if and only if* $\mathsf{App}_{\mathtt{e}}(\Gamma, \zeta, M)$ *and, moreover, for all* Γ' *and* N*,* $\mathsf{App}_{\mathtt{e}}(\Gamma', \vartheta, N)$ *implies* $\mathsf{App}_{\mathtt{e}}(\Gamma \uplus \Gamma', \zeta, MN)$*;*
(5) $\mathsf{Comp}_{\mathtt{i}}(\Gamma, \zeta, M)$*, if and only if* $\mathsf{App}_{\mathtt{i}}(\Gamma, \zeta, M)$ *and, moreover, for all* N*,* $\mathsf{App}_{\mathtt{i}}(\Gamma, \zeta, MN)$*;*
(6) $\mathsf{Comp}_{\mathtt{e}}(\Gamma, \vartheta, M)$*, if and only if* $\mathsf{App}_{\mathtt{e}}(\Gamma, \vartheta, M)$ *and, moreover, for all* Γ' *and* N*,* $\mathsf{App}_{\mathtt{e}}(\Gamma', \zeta, N)$ *implies* $\mathsf{App}_{\mathtt{e}}(\Gamma \uplus \Gamma', \vartheta, MN)$*;*
(7) $\mathsf{Comp}_{\mathtt{i}}(\Gamma, \vartheta, M)$*, if and only if* $\mathsf{App}_{\mathtt{i}}(\Gamma, \vartheta, M)$ *and, moreover, for all* N*,* $\mathsf{App}_{\mathtt{i}}(\Gamma, \vartheta, MN)$*;*
(8) $\mathsf{Comp}_{\varphi}(\Gamma, \alpha \to \beta, M)$*, if and only if* $\mathsf{App}_{\varphi}(\Gamma, \omega \to \omega, M)$*, and, moreover, for all* N *and* Γ' *such that* Γ *and* Γ' *are compatible bases,* $\mathsf{Comp}_{\varphi}(\Gamma', \alpha, N)$ *implies* $\mathsf{Comp}_{\varphi}(\Gamma \uplus \Gamma', \beta, MN)$*, when* $\varphi \in \{\mathtt{t}, \mathtt{w}\}$*;*
(9) $\mathsf{Comp}_{\varphi}(\Gamma, \alpha \to \beta, M)$*, if and only if, for all* Γ' *and* N*,* $\mathsf{Comp}_{\varphi}(\Gamma', \alpha, N)$ *implies* $\mathsf{Comp}_{\varphi}(\Gamma \uplus \Gamma', \beta, MN)$*, when* $\varphi \in \{\mathtt{h}, \mathtt{e}, \mathtt{i}\}$*;*
(10) $\mathsf{Comp}_{\varphi}(\Gamma, \alpha \wedge \beta, M)$ *if and only if* $\mathsf{Comp}_{\varphi}(\Gamma, \alpha, M)$ *and* $\mathsf{Comp}_{\varphi}(\Gamma, \beta, M)$*.*

The predicates App_{φ} and Comp_{φ} agree with the typing rule $(\leq_{\varphi})$ and depend only on the equivalence classes of terms modulo β-conversion.

Lemma 40 *(1) If $\alpha \leq_\varphi \beta$ and $\mathsf{Comp}_\varphi(\Gamma, \alpha, M)$, then $\mathsf{Comp}_\varphi(\Gamma, \beta, M)$*[5]*.*

(2) If $M =_\beta M'$, then $\mathsf{App}_\varphi(\Gamma, \alpha, M)$ if and only if $\mathsf{App}_\varphi(\Gamma, \alpha, M')$, and $\mathsf{Comp}_\varphi(\Gamma, \alpha, M)$ if and only if $\mathsf{Comp}_\varphi(\Gamma, \alpha, M')$.

(3) Let $z \notin FV(M)$ and $\Gamma' = \Gamma, z{:}\alpha$. Then $\mathsf{App}_\varphi(\Gamma, \alpha\to\beta, M)$, provided both $\mathsf{App}_\varphi(\Gamma', \beta, Mz)$ and $\mathsf{App}_\varphi(\Gamma, \omega\to\omega, M)$.

PROOF.

(1) By easy induction on the definition of $\leq_\varphi$.

(2) For App_φ, it suffices to observe that two β-convertible terms have the same approximants. For Comp_φ, we reason by induction on types.

(3) We consider only the case $\varphi = \mathtt{t}$. The proof of the other cases is similar and simpler. Note that $\mathsf{App}_\varphi(\Gamma, \omega\to\omega, M)$ is always true for $\varphi \in \{\mathtt{h}, \mathtt{e}, \mathtt{i}\}$, since in this case $\omega =_\varphi \omega\to\omega$.

Assume that $A \in \mathcal{A}_{\mathtt{t}}(Mz)$, $\Gamma' \vdash_{\mathtt{t}} A : \beta$, and $\Gamma \vdash_{\mathtt{t}} A' : \omega\to\omega$, for some $A' \in \mathcal{A}_{\mathtt{t}}(M)$. We must prove that there exists an $\widehat{A} \in \mathcal{A}_{\mathtt{t}}(M)$ such that $\Gamma \vdash_{\mathtt{t}} \widehat{A} : \alpha\to\beta$.

If $\beta =_{\mathtt{t}} \omega$, one has $\Gamma \vdash_{\mathtt{t}} A' : \alpha\to\omega$, since $\alpha \leq_{\mathtt{t}} \omega$; hence $\widehat{A} \equiv A'$. If M is β-convertible to an abstraction, then $\lambda z.Mz =_\beta M$ and we can choose $\widehat{A} \equiv \lambda z.A$.

If $A \equiv xA_1 \ldots A_n Z$, we take $\widehat{A} \equiv xA_1 \ldots A_n$. Indeed, since $\Gamma' \vdash_{\mathtt{t}} A : \beta$, it follows from Lemma 31(2c) that either $\beta \geq_{\mathtt{t}} \pi\gamma$ and $\Gamma' \vdash_{\mathtt{t}} \widehat{A} : \pi$, or $\Gamma' \vdash_{\mathtt{t}} \widehat{A} : \gamma\to\beta$, for some γ with $\Gamma' \vdash_{\mathtt{t}} Z : \gamma$. Notice that $z \notin FV(M)$ implies $z \notin \widehat{A}$, so we get either $\Gamma \vdash_{\mathtt{t}} \widehat{A} : \pi$, or $\Gamma \vdash_{\mathtt{t}} \widehat{A} : \gamma\to\beta$. The condition $\Gamma \vdash_{\mathtt{t}} A' : \omega\to\omega$ forbids $\Gamma \vdash_{\mathtt{t}} \widehat{A} : \pi$ by Lemma 32(2). As an approximant of z, the term Z is either z or $\perp$, and in both cases we must have $\alpha \leq_{\mathtt{t}} \gamma$. Thus we get $\Gamma \vdash_{\mathtt{t}} \widehat{A} : \alpha\to\beta$, as desired.

The case $A \equiv \perp A_1 \ldots A_n Z$ is excluded by Lemmas 31(2d) and 32(2). □

We can now show that computability implies approximability.

Lemma 41 *For all $\Gamma \in \mathcal{B}_{\bar{\varphi}}$, $\alpha \in \mathsf{T}_{\bar{\varphi}}$, $L_1, \ldots, L_n$ $(0 \leq n)$, and M:*

(1) $\mathsf{App}_\varphi(\Gamma, \alpha, xL_1 \ldots L_n) \Rightarrow \mathsf{Comp}_\varphi(\Gamma, \alpha, xL_1 \ldots L_n)$;

(2) $\mathsf{Comp}_\varphi(\Gamma, \alpha, M) \Rightarrow \mathsf{App}_\varphi(\Gamma, \alpha, M)$.

PROOF. We prove (1) and (2) by simultaneous induction on α.

- α is an atomic or an applicative type. Both (1) and (2) are true by definition of Comp_φ and the equalities $\vartheta =_{\mathtt{e}} \zeta\to\vartheta$, $\zeta =_{\mathtt{e}} \vartheta\to\zeta$, $\vartheta =_{\mathtt{i}} \omega\to\vartheta$, and $\zeta =_{\mathtt{i}} \omega\to\zeta$.

[5] The same property trivially holds for $\mathsf{App}_\varphi(\,)$.

- $\alpha \equiv \alpha_1 \to \alpha_2$.

(1) Assume $\mathsf{App}_\varphi(\Gamma, \alpha, xL_1 \dots L_n)$. Then there is an $A \in \mathcal{A}_\varphi(xL_1 \dots L_n)$ with $\Gamma \vdash_\varphi A : \alpha_1 \to \alpha_2$. This implies $\Gamma \vdash_\varphi A : \omega \to \omega$ by rule $(\leq_\varphi)$, so, in particular, $\mathsf{App}_\varphi(\Gamma, \omega \to \omega, xL_1 \dots L_n)$; this will be useful when $\varphi \in \{\mathtt{t}, \mathtt{w}\}$. Clearly, A can be taken of the form $xA_1 \dots A_n$, where A_i is an approximant of L_i $(i \leq n)$.

We need to show that $\mathsf{Comp}_\varphi(\Gamma, \alpha_1 \to \alpha_2, xL_1 \dots L_n)$. Let Γ' be compatible with Γ, and assume $\mathsf{Comp}_\varphi(\Gamma', \alpha_1, N)$, then $\mathsf{App}_\varphi(\Gamma', \alpha_1, N)$ follows by induction on (2). Hence, there is an approximant $B \in \mathcal{A}_\varphi(N)$ of type α_1 in the context Γ'. Then $AB \equiv xA_1 \dots A_n B$ is an approximant of $xL_1 \dots L_n N$, and $\Gamma \uplus \Gamma' \vdash AB : \alpha_2$. Thus $\mathsf{Comp}_\varphi(\Gamma \uplus \Gamma', \alpha_2, xL_1 \dots L_n N)$ follows by induction on (1).

(2) Suppose $\mathsf{Comp}_\varphi(\Gamma, \alpha_1 \to \alpha_2, M)$. Now $\mathsf{App}_\varphi(\Gamma, \omega \to \omega, M)$ follows by definition (this is necessary only for $\varphi \in \{\mathtt{t}, \mathtt{w}\}$ and will be used in the last of the following implications). Let $\Gamma' = \Gamma, z{:}\alpha_1$, where z is fresh. Since $\{z{:}\alpha_1\} \vdash_\varphi z : \alpha_1$, and $z \in \mathcal{A}_\varphi(z)$, we have $\mathsf{Comp}_\varphi(\{z{:}\alpha_1\}, \alpha_1, z)$ by induction on (1). Then we have

$$\begin{array}{lcl} \mathsf{Comp}_\varphi(\{z{:}\alpha_1\}, \alpha_1, z) \Rightarrow & & \text{(by definition of } \mathsf{Comp}_\varphi\text{)} \\ \mathsf{Comp}_\varphi(\Gamma', \alpha_2, Mz) & \Rightarrow & \text{(by induction on (2))} \\ \mathsf{App}_\varphi(\Gamma', \alpha_2, Mz) & \Rightarrow & \text{(by Lemma 40(3))} \\ \mathsf{App}_\varphi(\Gamma, \alpha_1 \to \alpha_2, M). & & \end{array}$$

- $\alpha \equiv \alpha_1 \wedge \alpha_2$.

(1) We need that if $\Gamma \vdash_\varphi A : \alpha_1 \wedge \alpha_2$, then $\Gamma \vdash_\varphi A : \alpha_1$ and $\Gamma \vdash_\varphi A : \alpha_2$, which follows by rule $(\leq_\varphi)$.

(2) We need that any two approximations have a common join (refinement) (see Lemma 12), and that if A' refines A, then A' has all the types of A in any basis (Lemma 32(1)). ◻

Lemma 42 *Let* $\Gamma = \{x_1{:}\beta_1, \dots, x_n{:}\beta_n\} \in \mathcal{B}_{\bar\varphi}$ *and* $\Gamma \vdash_\varphi M : \alpha$. *Assume, for* $i \leq n$, $\mathsf{Comp}_\varphi(\Gamma_i, \beta_i, N_i)$. *Define* $\Gamma' = \biguplus_{i=1}^n \Gamma_i$. *Then* $\mathsf{Comp}_\varphi(\Gamma', \alpha, M[\overline{N/x}])$, *where* $[\overline{N/x}]$ *is shorthand for* $[N_1/x_1, \dots, N_n/x_n]$.

PROOF. By induction on the derivation for $\Gamma \vdash_\varphi M : \alpha$. Cases (*Ax*) and (ω) are immediate. Cases ($\to$*E*) and ($\wedge$*I*) follow by induction. Case $(\leq_\varphi)$ follows by induction and Lemma 40(1). For $\varphi = \mathtt{t}$, case $(\beta{-}exp)$ follows by induction and Lemma 40(2).

For ($\to$*I*), let $M \equiv \lambda y.P$ and $\alpha \equiv \alpha_1 \to \alpha_2$, then $\Gamma, y{:}\alpha_1 \vdash_\varphi P : \alpha_2$. Since $\lambda y.\bot$ is an approximant of $(\lambda y.P)[\overline{N/x}]$ of type $\omega \to \omega$, we have $\mathsf{App}_\varphi(\Gamma, \omega \to \omega, M[\overline{N/x}])$ (this is useful only for $\varphi \in \{\mathtt{t}, \mathtt{w}\}$).

Suppose $\mathsf{Comp}_\varphi(\Gamma'', \alpha_1, Q)$. Then, by induction

$$\mathsf{Comp}_\varphi(\Gamma' \uplus \Gamma'', \alpha_2, P[Q/y, \overline{N/x}]).$$

We can assume, without loss of generality, that $y \notin FV(N_1 \dots N_n)$ and, therefore,

$$P[Q/y, \overline{N/x}] \equiv P[\overline{N/x}][Q/y] \text{ and } (\lambda y.P[\overline{N/x}])Q \equiv ((\lambda y.P)[\overline{N/x}])Q.$$

By the invariance of computability under β-conversion (Lemma 40(2)), we have $\mathsf{Comp}_\varphi(\Gamma' \uplus \Gamma'', \alpha_2, ((\lambda y.P)[\overline{N/x}])Q)$, and hence

$$\mathsf{Comp}_\varphi(\Gamma', \alpha_1 \to \alpha_2, (\lambda y.P)[\overline{N/x}]).$$

For $\vdash_{\mathtt{t}}$ we need to consider also rules (ω*app*) and (*app*). We will give the proof for (*app*), the proof for (ω*app*) is similar and simpler. For rule (*app*), assume $M \equiv PQ$ and $\alpha \equiv \pi\gamma$. We get $\mathsf{Comp}_{\mathtt{t}}(\Gamma', \pi, P[\overline{N/x}])$ and $\mathsf{Comp}_{\mathtt{t}}(\Gamma', \gamma, Q[\overline{N/x}])$ by induction. Then, by Lemma 41(2), $\mathsf{App}_{\mathtt{t}}(\Gamma', \pi, P[\overline{N/x}])$ and $\mathsf{App}_{\mathtt{t}}(\Gamma', \gamma, Q[\overline{N/x}])$. This means $\Gamma' \vdash_{\mathtt{t}} A : \pi$ for some $A \in \mathcal{A}_{\mathtt{t}}(P[\overline{N/x}])$ and $\Gamma' \vdash_{\mathtt{t}} A' : \gamma$ for some $A' \in \mathcal{A}_{\mathtt{t}}(Q[\overline{N/x}])$. Notice that, by Lemma 31(2b), A cannot be an abstraction, so $AA' \in \mathcal{A}_{\mathtt{t}}(M[\overline{N/x}])$. Moreover, we derive $\Gamma' \vdash_{\mathtt{t}} AA' : \pi\gamma$, so we conclude $\mathsf{Comp}_{\mathtt{t}}(\Gamma', \pi\gamma, M[\overline{N/x}])$. □

We can now prove the approximation theorem.

Theorem 43 (Approximation Theorem) *$\Gamma \vdash_\varphi M : \alpha$ if and only if there is $A \in \mathcal{A}_\varphi(M)$ such that $\Gamma \vdash_\varphi A : \alpha$.*

PROOF.

($\Rightarrow$) Since $\mathsf{App}_\varphi(\{x{:}\beta\}, \beta, x)$ holds for any x and β, then, by Lemma 41(1), we have $\mathsf{Comp}_\varphi(\{x{:}\beta\}, \beta, x)$. We can apply Lemma 42 for the identity substitution to obtain $\mathsf{Comp}_\varphi(\Gamma, \alpha, M)$. We conclude using Lemma 41(2).

($\Leftarrow$) Without loss of generality we can assume that $A \equiv (M)^{(h)}_\varphi$, for some h.

- For $\varphi \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}\}$, this implies, by Definition 4, that there is M' such that $M \to_\beta M'$ and A is obtained from M' by replacing some subterm by $\perp$. So we get $\Gamma \vdash_\varphi M' : \alpha$, and $\Gamma \vdash_\varphi M : \alpha$ follows from rule (β*−exp*), which is admissible in $\vdash_\varphi$ for $\varphi \in \{\mathtt{w}, \mathtt{h}\}$ by Theorem 34(3).
- For $\varphi = \mathtt{e}$, by Definition 4, there is M' such that $M \to_\beta M'$ and A is obtained from M' by replacing some subterm by $\perp$ and by η-reduction. Then, since types are preserved by η-expansion in $\vdash_{\mathtt{e}}$ as proved in Theorem 34(4), $\Gamma \vdash_{\mathtt{e}} M' : \alpha$. We conclude as in previous case.
- For $\varphi = \mathtt{i}$, by Definition 5, there is M' such that $M \to_\beta M'$ and A is obtained from M' by:
  - · replacing some subterm by $\perp$;

· η-reduction;
· replacing some subterm N such that $\mathcal{T}_{\mathtt{i}}(N) \geq_\eta x$ by x.

So we get $\Gamma \vdash_{\mathtt{i}} M' : \alpha$, by Theorem 34(4) and by Theorem 38(2). □

5 Correspondence between trees and typings

In this section we will present the main result of the paper, namely that our type assignment systems can be used to analyze the observational behavior represented by trees. As recalled in the introduction, similar results are present in the literature for particular notions of tree.

Ronchi della Rocca [24] proved that two terms have the same Böhm tree if and only if they have the same set of types in the standard intersection type discipline [5]. The proof of [24] is based on the notion of principal type of an approximate normal form, which is a type completely describing the approximate normal form. Principal types (as defined in [11] and used in [24]) need an infinity of type variables and this agrees with the type syntax of [5]. Another related paper is [16]: it proves that two terms have the same Lévy-Longo tree [22] if and only if they have the same set of types in the type discipline with union and intersection of [13]. Also [16] uses the notion of principal types, but it gets rid of type variables by replacing them by suitable constant types which depend on the terms involved. Lastly, [7] proves this correspondence in the case of Berarducci trees for a type assignment system quite similar to $\vdash_{\mathtt{t}}$ by taking advantage from the presence of applicative types.

In the following we shall provide an (almost) uniform proof for a theorem which considers other trees besides those of the results recalled above. More precisely, we shall prove that $\vdash_\varphi$ derives the same types for two terms M, N if and only if M, N have the same φ-trees.

In order to prove this property, we follow an approach similar to [16] and to [7] in that we do not allow an infinite set of type variables. The expressive power needed for our purposes and that could be provided by an infinity of type variables can be obtained instead by defining, as we shall do, an infinite set of constant types. These constants will also allow to define the characteristic pairs $\langle$*basis; type*$\rangle$ for approximate normal forms.

The key idea is that characteristic pairs give sufficient information to discriminate between approximate normal forms obtained by pruning (in a suitable way) different trees.

We introduce three different sets of type constants, one for each set of types ($\mathsf{T}_{\mathtt{t}}$, $\mathsf{T}_{\mathtt{wh}}$ and $\mathsf{T}_{\mathtt{ei}}$). It is easy to verify that each of these constants belong to the corresponding set of types.

Definition 44 *(1) Let $\theta = (\omega\omega{\to}\omega{\to}\omega)\wedge((\omega{\to}\omega){\to}\omega\omega)$. We define ϕ_0 as the type $\omega\theta$ and, for $i \geq 0$, $\phi_{i+1} \equiv \omega(\phi_i\theta)$.*

(2) Define $\psi_i^{(n)} = (\zeta{\to}\omega^i{\to}\zeta{\to}\omega^{n-i}{\to}\zeta)\wedge\zeta$, for all $i \leq n$.

(3) Define $\chi_i^{(n)} = \zeta{\to}\vartheta^i{\to}\zeta{\to}\vartheta^{n-i}{\to}\zeta{\to}\vartheta\wedge\zeta$, for all $i \leq n$.

The following lemma states that for some properties we shall need in our proofs, the type constants defined above behave as type variables.

Lemma 45 *(1) If $\phi_i\alpha_1\ldots\alpha_m \leq_{\mathtt{t}} \phi_j\beta_1\ldots\beta_n$ and $\alpha_l \not\leq_{\mathtt{t}} \theta$, $\alpha_l \not\leq_{\mathtt{t}} \phi_k\theta$, where $1 \leq l \leq m$ and $k \geq 0$, then $i = j$, $m = n$ and $\alpha_l \leq_{\mathtt{t}} \beta_l$, for $1 \leq l \leq m$.*

(2) Let $\varphi \in \{\mathtt{w}, \mathtt{h}\}$, then

$$\bigwedge_{i\in I}(\alpha_1^{(i)}{\to}\ldots{\to}\alpha_{n_i}^{(i)}{\to}\psi_i^{(n)}) \leq_\varphi \beta_1{\to}\ldots{\to}\beta_m{\to}\psi_j^{(n)},$$

and $n_i \leq n$, for all $i \in I$, imply $j \in I$, $n_j = m$ and $\beta_l \leq_\varphi \alpha_l^{(j)}$, for $1 \leq l \leq m$.

(3) Let $\varphi \in \{\mathtt{e}, \mathtt{i}\}$, then

$$\bigwedge_{i\in I}(\alpha_1^{(i)}{\to}\ldots{\to}\alpha_{n_i}^{(i)}{\to}\zeta^n{\to}\chi_i^{(n)}) \leq_\varphi \beta_1{\to}\ldots{\to}\beta_m{\to}\zeta^{n-k}{\to}\chi_j^{(n)}$$

implies $j \in I$, $n_j = m - k$, $\beta_h \leq_\varphi \alpha_h^{(j)}$, for $1 \leq h \leq n_j$, and $\beta_h \leq_\varphi \zeta$, for $n_j + 1 \leq h \leq m$.

PROOF.

(1) We first show that $m = n$. Assuming $m > n$, by Lemma 25(3), we get $\phi_i\alpha_1\ldots\alpha_{m-n} \leq_{\mathtt{t}} \phi_j$, which implies $\alpha_{m-n} \leq_{\mathtt{t}} \theta$, whenever $j = 0$, and $\alpha_{m-n} \leq_{\mathtt{t}} \phi_{j-1}\theta$, whenever $j > 0$. Both inequalities are false by assumption. Assuming $m < n$, we get $\phi_i \leq_{\mathtt{t}} \phi_j\beta_1\ldots\beta_{n-m}$, which implies $\omega \leq_{\mathtt{t}} \phi_j\beta_1\ldots\beta_{n-m-1}$, which is false.

If $m = n$, we have $\alpha_l \leq_{\mathtt{t}} \beta_l$, for $1 \leq l \leq m$, and $\phi_i \leq_{\mathtt{t}} \phi_j$. If $i = 0$ and $j > 0$, we get $\theta \leq_{\mathtt{t}} \phi_{j-1}\theta$. If $i > 0$ and $j = 0$, we get $\phi_{i-1}\theta \leq_{\mathtt{t}} \theta$. Both inequalities are false since arrow types are incomparable with applicative types. If $i > 0$ and $j > 0$, we get $\phi_{i-1}\theta \leq_{\mathtt{t}} \phi_{j-1}\theta$, i.e., $\phi_{i-1} \leq_{\mathtt{t}} \phi_{j-1}$. We conclude that $i = j$.

(2) Note that

$$\bigwedge_{i\in I}(\alpha_1^{(i)}{\to}\ldots{\to}\alpha_{n_i}^{(i)}{\to}\psi_i^{(n)}) \leq_\varphi \beta_1{\to}\ldots{\to}\beta_m{\to}\psi_j^{(n)}$$

implies

$$\bigwedge_{i\in I}(\alpha_1^{(i)}{\to}\ldots{\to}\alpha_{n_i}^{(i)}{\to}\psi_i^{(n)}) \leq_\varphi \beta_1{\to}\ldots{\to}\beta_m{\to}\zeta{\to}\omega^j{\to}\zeta{\to}\omega^{n-j}{\to}\zeta.$$

By Lemma 26(1) and clause (c) of Definition 23, for some i either

$$\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta \leq_\varphi \beta_1\to\ldots\to\beta_m\to\zeta\to\omega^j\to\zeta\to\omega^{n-j}\to\zeta \tag{1}$$

or

$$\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta\to\omega^i\to\zeta\to\omega^{n-i}\to\zeta \leq_\varphi \beta_1\to\ldots\to\beta_m\to\zeta\to\omega^j\to\zeta\to\omega^{n-j}\to\zeta. \tag{2}$$

The type inclusion (1) is impossible, since, by Lemma 26(2), it requires $n_i = m+n+2$. To satisfy (2), we get $n_i+n = m+n$, i.e., $n_i = m$. Moreover, $i = j$: in fact, assuming $i \neq j$ we obtain, by Lemma 26(2), $\omega \leq_\varphi \zeta$, which is false. The conclusion follows from Lemma 26(2).

(3) Notice that

$$\bigwedge_{i\in I}(\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta^n\to\chi_i^{(n)}) \leq_\varphi \beta_1\to\ldots\to\beta_m\to\zeta^{n-k}\to\chi_j^{(n)}$$

implies, by Lemma 26(1), for some i,

$$\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta^n\to\chi_i^{(n)} \leq_\varphi \beta_1\to\ldots\to\beta_m\to\zeta^{n-k}\to\chi_j^{(n)}.$$

By Definition 44(3), we get

$$\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta^n\to\zeta\to\vartheta^i\to\zeta\to\vartheta^{n-i}\to\zeta\to\vartheta\wedge\zeta \leq_\varphi \beta_1\to\ldots\to\beta_m\to\zeta^{n-k}\to\zeta\to\vartheta^j\to\zeta\to\vartheta^{n-j}\to\zeta\to\vartheta\wedge\zeta. \tag{3}$$

To satisfy (3), we get $n_i = m-k$, and $i = j$. In fact, assuming $n_i \neq m-k$ or $i \neq j$ we obtain, by Lemma 26(2), $\vartheta \leq_\varphi \zeta$, or $\zeta \leq_\varphi \vartheta$, which are both false. The conclusion follows from Lemma 26(2). $\square$

We need to consider special kinds of bases which allow to distinguish occurrences of different variables or even different occurrences of the same variable. More precisely, in the presence of applicative types it suffices to give different types to occurrences of different variables, but in all other cases we need to give also different types to different occurrences of the same variable.

Definition 46 *(1) We define* $\Gamma_{\mathtt{t}} \in \mathcal{B}_{\mathtt{t}}$ *as the basis* $\{x_n{:}\phi_n \mid n \in \mathbb{N}\}$.

(2) A basis $\Gamma \in \mathcal{B}_{\mathtt{wh}}$ *is a* special basis (of degree n) *if each type declaration in* Γ *has the form* $x{:}\bigwedge_{i\in I}(\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\psi_i^{(n)})$, *where* $n_i \leq n$, *for all* $i \in I$, *and, moreover, each* $\psi_i^{(n)}$ *occurs only once as last type.*

(3) $\Gamma \in \mathcal{B}_{\mathtt{ei}}$ *is a* generalized special basis (of degree n) *if each type declaration in* Γ *has the form* $x{:}\zeta$ *or* $x{:}\bigwedge_{i\in I}(\alpha_1^{(i)}\to\ldots\to\alpha_{n_i}^{(i)}\to\zeta^n\to\chi_i^{(n)})$, *where* $n_i \leq n$, *for all* $i \in I$, *and, moreover, each* $\chi_i^{(n)}$ *occurs only once as last type.*

Notice that $\Gamma_{\mathtt{t}}$ contains only applicative types, while special bases and generalized special bases contain only arrow types and atomic types. The feature of all these bases is that when we deduce from them a type which behaves like a variable for an approximate normal form, we can argue that the approximate normal form has a fixed shape, and that its components have fixed types.

Lemma 47 *(1) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} x_i A_1 \dots A_n : \alpha$ and $\alpha \neq_{\mathtt{t}} \omega$, then α is an applicative type.*

(2) For any approximate normal form A neither $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \theta$, nor $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \phi_i\theta$, for $i \geq 0$.

(3) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \phi_i$, then $A \equiv x_i$.

PROOF.

(1) By induction on n. If $n = 0$, the thesis follows from Lemmas 31(2a) and 25(2), since all types in $\Gamma_{\mathtt{t}}$ are applicative. Otherwise, by induction, $x_i A_1 \dots A_{n-1}$ has only applicative types, and we obtain the thesis by Lemma (31)(2c).

(2) Assume $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \theta$. Then (1) forces A to be of the form $\lambda x.A'$. Recalling the definition of θ, we get, by Lemma 31(2b), $\Gamma_{\mathtt{t}}, x{:}\omega\omega \vdash_{\mathtt{t}} A' : \omega\to\omega$ and $\Gamma_{\mathtt{t}}, x{:}\omega\to\omega \vdash_{\mathtt{t}} A' : \omega\omega$. Then, since we can assign both an arrow type and an applicative type to A', Lemma 31(1) and (2) imply $A' \equiv yB_1 \dots B_n$, for some $y, n, B_1, \dots, B_n$. Hence, we get $\Gamma_{\mathtt{t}}, x{:}\omega\omega \vdash_{\mathtt{t}} y : \alpha_1\to\dots\to\alpha_n\to\omega\to\omega$, for some $\alpha_1, \dots, \alpha_n$. This is impossible by Lemma 31(2a) and Lemma 25(2), since all types in $\Gamma_{\mathtt{t}}, x{:}\omega\omega$ are applicative.

Assume $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \phi_i\theta$, for some i. By Lemma 31(2), we have either $A \equiv \bot A_1 \dots A_n$ or $A \equiv x_j A_1 \dots A_n$, for some $x_j, A_1, \dots, A_n$. If $n = 0$ then, by Lemma 31(2a), we get either $\omega \leq_{\mathtt{t}} \phi_i\theta$ or $\phi_j \leq_{\mathtt{t}} \phi_i\theta$, which are both false, the second one by Lemma 45(1). For $n > 0$, we have necessarily $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A_n : \theta$, which is impossible by the above.

(3) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \phi_i$ Lemma 31(2) implies either (a) $A \equiv \bot A_1 \dots A_n A'$ or (b) $A \equiv xA_1 \dots A_n$, for some $A_1, \dots, A_n, A', x$.

  (a) Then $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A' : \theta$, if $i = 0$, and $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A' : \phi_{i-1}\theta$, if $i > 0$, by Lemma 31(2d). Both cases are impossible by (2).

  (b) If $n > 0$, then, by Lemma 31(2c), either we must deduce an arrow type for x from $\Gamma_{\mathtt{t}}$ – which is impossible by (1) – or A' must have type θ or $\phi_{i-1}\theta$. We conclude that $n = 0$ and $A \equiv x_i$, because if $A \equiv x_j$ we have $\phi_i \leq_{\mathtt{t}} \phi_j$ by Lemma 31(2a), which implies $i = j$ by Lemma 45(1). □

Lemma 48 *(1) Let $\varphi \in \{\mathtt{w}, \mathtt{h}\}$. If $A \in \mathcal{A}_\varphi$, $\Gamma = \Gamma' \uplus \{x{:}\alpha_1\to\dots\to\alpha_m\to\psi_i^{(n)}\}$ is a special basis of degree n, and $\Gamma \vdash_\varphi A : \psi_i^{(n)}$, then $A \equiv xA_1 \dots A_m$ and, for $1 \leq j \leq m$, $\Gamma \vdash_\varphi A_j : \alpha_j$.*

(2) Let $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. If $A \in \mathcal{A}_\varphi^{(n)}$, and $\Gamma \uplus \{x{:}\alpha_1\to\dots\to\alpha_m\to\zeta^n\to\chi_i^{(n)}\}$ is a generalized special basis of degree n, and $\Gamma \vdash_\varphi A : \zeta^n\to\chi_i^{(n)}$, then, for some

$k \geq 0$, there are $B_1, \ldots, B_k$, such that $A \equiv \lambda y_1 \ldots y_k.xA_1 \ldots A_m B_1 \ldots B_k$, with $x \notin \{y_1, \ldots, y_k\}$, $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi A_j : \alpha_j$, for $1 \leq j \leq m$, and $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi B_l : \zeta$, for $1 \leq l \leq k$.

PROOF.

(1) From $\Gamma \vdash_\varphi A : \psi_i^{(n)}$ we get $\Gamma \vdash_\varphi A : \zeta$, so, by Lemma 33(3), A cannot be an abstraction. Assume $A \equiv yA_1 \ldots A_p$. Then, by Lemma 33(4), we have that $\Gamma \vdash_\varphi yA_1 \ldots A_p : \psi_i^{(n)}$ requires both $\Gamma \vdash_\varphi y : \beta_1 \to \ldots \to \beta_p \to \psi_i^{(n)}$ as well as $\Gamma \vdash_\varphi A_j : \beta_j$, for some $\beta_1, \ldots, \beta_p$, $(1 \leq j \leq p)$. By definition of special bases, the statement with subject y must have a predicate like $\bigwedge_{l \in L}(\alpha_1^{(l)} \to \ldots \to \alpha_{n_l}^{(l)} \to \psi_l^{(n)})$. By Lemma 33(2),

$$\bigwedge_{l \in L}(\alpha_1^{(l)} \to \ldots \to \alpha_{n_l}^{(l)} \to \psi_l^{(n)}) \leq_\varphi \beta_1 \to \ldots \to \beta_p \to \psi_i^{(n)}.$$

By Lemma 45(2), this implies $i \in L$, $n_i = p$ and $\beta_j \leq_\varphi \alpha_j^{(i)}$, for $1 \leq j \leq p$. By definition of special basis, $\psi_i^{(n)}$ can occur only once as last type, hence we conclude $x \equiv y$, $p = m$ and $\beta_j \leq \alpha_j$, for $1 \leq j \leq p$.

(2) Let $A \equiv \lambda y_1 \ldots y_k.zA_1 \ldots A_p$, where $k \leq n$, since $A \in \mathcal{A}_\varphi^{(n)}$. Then, by Lemma 33(3), $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi zA_1 \ldots A_p : \zeta^{n-k} \to \chi_i^{(n)}$. Now, to obtain $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi zA_1 \ldots A_p : \zeta^{n-k} \to \chi_i^{(n)}$, by Lemma 33(4) we need both

$$\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi z : \beta_1 \to \ldots \to \beta_p \to \zeta^{n-k} \to \chi_i^{(n)}$$

and

$$\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi A_j : \beta_j,$$

for some $\beta_1, \ldots, \beta_p$ $(1 \leq j \leq p)$. With a proof similar to that of the previous point, using Lemma 45(3) instead of Lemma 45(2), we conclude $x \equiv z$, $p = m + k$, $\beta_j \leq_\varphi \alpha_j$, for $1 \leq j \leq m$, and $\beta_j \leq_\varphi \zeta$, for $m + 1 \leq j \leq p$. □

We now associate to each approximate normal form $A \in \mathcal{A}_\varphi$ a basis $\Gamma \in \mathcal{B}_{\bar{\varphi}}$ and a type $\gamma \in \mathsf{T}_{\bar{\varphi}}$. We call the pair $\langle \Gamma; \gamma \rangle$ the φ-characteristic pair of A.

Definition 49 *Let $A \in \mathcal{A}_{\mathtt{t}}$.*

(1) The $\mathtt{t}$-characteristic type of A, $\mathsf{ct}_{\mathtt{t}}(A)$, *is defined as follows.*
   (a) $\mathsf{ct}_{\mathtt{t}}(\lambda x_i.A) = \phi_i \to \mathsf{ct}_{\mathtt{t}}(A)$,
   (b) $\mathsf{ct}_{\mathtt{t}}(\bot A_1 \ldots A_n) = \omega \mathsf{ct}_{\mathtt{t}}(A_1) \ldots \mathsf{ct}_{\mathtt{t}}(A_n)$,
   (c) $\mathsf{ct}_{\mathtt{t}}(x_i A_1 \ldots A_n) = \phi_i \mathsf{ct}_{\mathtt{t}}(A_1) \ldots \mathsf{ct}_{\mathtt{t}}(A_n)$.

(2) The $\mathtt{t}$-characteristic pair of A, $\mathsf{cp}_{\mathtt{t}}(A)$, *is* $\langle \Gamma_{\mathtt{t}}; \mathsf{ct}_{\mathtt{t}}(A) \rangle$.

It is easy to verify that $\Gamma_{\texttt{t}} \vdash_{\varphi} A : \mathsf{ct}_{\texttt{t}}(A)$.

Definition 50 *Let $A \in \mathcal{A}_{\varphi}^{(n)}$, for $\varphi \in \{\texttt{w}, \texttt{h}, \texttt{e}, \texttt{i}\}$. The φ-*characteristic pair of degree n of A, *$\mathsf{pp}_{\varphi}^{(n)}(A)$, is defined as follows.*

(1) $\mathsf{pp}_{\varphi}^{(n)}(\bot) = \langle \emptyset; \omega \rangle$

(2) If $\mathsf{pp}_{\varphi}^{(n)}(A) = \langle \Gamma, x{:}\beta; \alpha \rangle$, then $\mathsf{pp}_{\varphi}^{(n)}(\lambda x.A) = \langle \Gamma; \beta \to \alpha \rangle$.

(3) If $\mathsf{pp}_{\varphi}^{(n)}(A) = \langle \Gamma; \alpha \rangle$ and x does not occur in Γ, then $\mathsf{pp}_{\varphi}^{(n)}(\lambda x.A)$ is equal to $\langle \Gamma; \omega \to \alpha \rangle$.

(4) For $\varphi \in \{\texttt{w}, \texttt{h}\}$: assume $\mathsf{pp}_{\varphi}^{(n)}(A_i) = \langle \Gamma_i; \alpha_i \rangle$, for $i \leq k$, and let

$$\Gamma = \biguplus_{i=1}^{k} \Gamma_i \uplus \{x{:}\alpha_1 \to \ldots \to \alpha_k \to \psi_j^{(n)}\}$$

be a special basis of degree n. Then

$$\mathsf{pp}_{\varphi}^{(n)}(x A_1 \ldots A_k) = \langle \Gamma; \psi_j^{(n)} \rangle.$$

In particular, when $k = 0$, we obtain $\mathsf{pp}_{\varphi}^{(n)}(x) = \langle \{x{:}\psi_j^{(n)}\}; \psi_j^{(n)} \rangle$.

(5) For $\varphi \in \{\texttt{e}, \texttt{i}\}$: assume $\mathsf{pp}_{\varphi}^{(n)}(A_i) = \langle \Gamma_i; \alpha_i \rangle$, for $i \leq k$, and let

$$\Gamma = \biguplus_{i=1}^{k} \Gamma_i \uplus \{x{:}\alpha_1 \to \ldots \to \alpha_k \to \zeta^n \to \chi_j^{(n)}\}$$

be a generalized special basis of degree n, then

$$\mathsf{pp}_{\varphi}^{(n)}(x A_1 \ldots A_k) = \langle \Gamma; \zeta^n \to \chi_j^{(n)} \rangle.$$

In particular, when $k = 0$, we get $\mathsf{pp}_{\varphi}^{(n)}(x) = \langle \{x{:}\zeta^n \to \chi_j^{(n)}\}; \zeta^n \to \chi_j^{(n)} \rangle$.

Notice that the φ-characteristic pair of an approximant, as defined above, is not uniquely determined. In fact, to simplify, we assumed that the definition comes implicitly equipped with a procedure for choosing names of type variables (the 'j'-subscripts), whenever needed. This choice can always be made in such a way the definition is uniquely determined and sound.

It is easy to verify that, if $A \in \mathcal{A}_{\varphi}^{(n)}$ and $\mathsf{pp}_{\varphi}^{(n)}(A) = \langle \Gamma; \alpha \rangle$, then $\Gamma \vdash_{\texttt{t}} A : \alpha$, for $\varphi \in \{\texttt{w}, \texttt{h}, \texttt{e}, \texttt{i}\}$.

We can now prove that, in all cases, if the φ-characteristic pair of A is $\langle \Gamma; \gamma \rangle$ and $\Gamma \vdash_{\varphi} B : \gamma$ (and some conditions on the number of symbols of B or of $(B)_{\varphi}^{(h)}$, where h is the height of the tree of A, hold), then $A \sqsubseteq_{\varphi} B$, with $\sqsubseteq_{\varphi}$ as defined in Definition 15.

Lemma 51 *(1) If $\Gamma_{\texttt{t}} \vdash_{\texttt{t}} A : \alpha$ then $\alpha \not\leq_{\texttt{t}} \theta$, $\alpha \not\leq_{\texttt{t}} \phi_i \theta$, for all $i \geq 0$.*

(2) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \pi\alpha$ and $\phi_i \not\leq_{\mathtt{t}} \pi\alpha$, for all i, then $A \equiv A_1A_2$ with $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A_1 : \pi$ and $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A_2 : \alpha$.
(3) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \phi_i \to \alpha$, then $A \equiv \lambda x_i.A'$ and $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A' : \alpha$.
(4) If $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B : \mathsf{ct}_{\mathtt{t}}(A)$ then $A \sqsubseteq_{\mathtt{t}} B$.

PROOF.

(1) Assume, reasoning towards a contradiction, that $\alpha \leq_{\mathtt{t}} \beta$, where either $\beta = \theta$, or $\beta = \phi_i\theta$. Then we get $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} A : \beta$, which is impossible by Lemma 47(2).
(2) The approximate normal form A cannot be a variable by Lemma 31(2a), since $\phi_i \not\leq_{\mathtt{t}} \pi\alpha$, for all i. It cannot be an abstraction by Lemma 31(2b). Therefore A must be an application, i.e., $A \equiv A_1A_2$. Notice that A_1 cannot have an arrow type by Lemmas 31(2) and 47(1). So we get the thesis using Lemma 31(2c), and (2d).
(3) By Lemma 47(1) and Lemma 31(2d), A can neither be a variable nor an application. So $A \equiv \lambda x_i.A'$ and we get the thesis by Lemma 31(2b).
(4) Suppose $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B : \mathsf{ct}_{\mathtt{t}}(A)$. We prove that $A \sqsubseteq_{\mathtt{t}} B$ by induction on the structure of A.
- The crucial case is when $A \equiv \bot A_1 \dots A_n$ ($n \geq 0$). Then $\mathsf{ct}_{\mathtt{t}}(A)$ has the form $\omega\alpha_1 \dots \alpha_n$. If B has type $\mathsf{ct}_{\mathtt{t}}(A)$ in the basis $\Gamma_{\mathtt{t}}$, then, by (2) and Lemma 31(2), B must be of the form $xB_1 \dots B_m$ or $xB_1 \dots B_pB'_1 \dots B'_n$ or $\bot B_1 \dots B_pB'_1 \dots B'_n$ where $0 \leq m < n$ and $0 \leq p$. In the first case, by (2), for $i = 1, \dots, m$, B_i has type α_{n-m+i} in $\Gamma_{\mathtt{t}}$. By induction, for $i = 1, \dots, m$, $A_{n-m+i} \sqsubseteq_{\mathtt{t}} B_i$. Since $\bot A_1 \dots A_{n-m} \sqsubseteq_{\mathtt{t}} x$ we infer that $A \sqsubseteq_{\mathtt{t}} B$. In the remaining cases, for $i = 1, \dots, n$, B'_i has type α_i in $\Gamma_{\mathtt{t}}$. By induction, for $i = 1, \dots, n$, $A_i \sqsubseteq_{\mathtt{t}} B'_i$. Since $\bot \sqsubseteq_{\mathtt{t}} xB_1 \dots B_p$ and $\bot \sqsubseteq_{\mathtt{t}} \bot B_1 \dots B_p$, we conclude again that $A \sqsubseteq_{\mathtt{t}} B$.
- If $A \equiv x_iA_1 \dots A_n$ ($n \geq 0$), then $\mathsf{ct}_{\mathtt{t}}(A)$ has the form $\phi_i\alpha_1 \dots \alpha_n$. If B has type $\mathsf{ct}_{\mathtt{t}}(A)$ in the basis $\Gamma_{\mathtt{t}}$, then by (2) and Lemma 31(2), B must be either of the form (a) $x_jB_1 \dots B_m$ or (b) $\bot B_1 \dots B_m$.
  (a) Then $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B : \phi_i\alpha_1 \dots \alpha_n$ implies $\phi_j\beta_1 \dots \beta_m \leq_{\mathtt{t}} \phi_i\alpha_1 \dots \alpha_n$, by Lemma 31(2c), for some β_l such that $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B_l : \beta_l$ ($1 \leq l \leq m$). Notice that $\beta_l \not\leq_{\mathtt{t}} \theta$ and $\beta_l \not\leq_{\mathtt{t}} \phi_k\theta$ by (1). ¿From this, by Lemma 45(1), we infer $i = j$, $m = n$ and $\beta_l \leq_{\mathtt{t}} \alpha_l$ for $1 \leq l \leq n$. Hence, by induction, $A_l \sqsubseteq_{\mathtt{t}} B_l$, for $1 \leq l \leq n$, and this implies $A \sqsubseteq_{\mathtt{t}} B$.
  (b) Then $\omega\beta_1 \dots \beta_m \leq_{\mathtt{t}} \phi_i\alpha_1 \dots \alpha_n$, for some β_l ($1 \leq l \leq m$) such that $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B_l : \beta_l$. This case is impossible. In fact, if $m > n$ we get $\beta_{m-n} \leq_{\mathtt{t}} \theta$, whenever $i = 0$ and $\beta_{m-n} \leq_{\mathtt{t}} \phi_{i-1}\theta$, whenever $i > 0$. If $m \leq n$, we get $\omega \leq_{\mathtt{t}} \phi_i\alpha_1 \dots \alpha_{n-m}$.
- If $A \equiv \lambda x_i.A'$, then $\mathsf{ct}_{\mathtt{t}}(A)$ has the form $\phi_i \to \alpha$. If B has type $\mathsf{ct}_{\mathtt{t}}(A)$ in $\Gamma_{\mathtt{t}}$ then B must be an abstraction $\lambda y.B'$ by Lemma 51(3) and by a renaming of bound variables we can assume $y \equiv x_i$. Then $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} B' : \alpha$ and induction applies.
- If $A \equiv \bot$ there is nothing to prove. ◻

Lemma 52 *Let $\varphi \in \{\mathtt{w}, \mathtt{h}\}$, $A \in \mathcal{A}^{(n)}_\varphi$ and $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Gamma; \alpha \rangle$. Then $B \in \mathcal{A}_\varphi$ and $\Gamma \vdash_\varphi B : \alpha$ imply $A \sqsubseteq_\varphi B$.*

PROOF. We prove by induction on A a stronger claim:

Let $A \in \mathcal{A}^{(n)}_\varphi$, $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta; \alpha \rangle$ and Γ be a special basis of degree n which contains Δ. Then $B \in \mathcal{A}_\varphi$ and $\Gamma \vdash_\varphi B : \alpha$ imply $A \sqsubseteq_\varphi B$.

The desired property is then obtained by taking $\Gamma = \Delta$.

- $A \equiv \lambda x.A'$. Then $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta; \beta \to \alpha \rangle$ and $\langle \Delta, x{:}\beta; \alpha \rangle = \mathsf{pp}(A')$, if $\beta \neq_\varphi \omega$, or $\langle \Delta; \alpha \rangle = \mathsf{pp}(A')$, otherwise. If $B \equiv \lambda x.B'$, then $\Gamma \vdash_\varphi B : \beta \to \alpha$ implies $\Gamma, x{:}\beta \vdash_\varphi B' : \alpha$ and we apply the induction hypothesis. If $B \equiv yB_1 \dots B_m$, then $\Gamma \vdash_\varphi B : \beta \to \alpha$ implies $\Gamma \vdash_\varphi \lambda z.Bz : \beta \to \alpha$, where $z \notin FV(B)$. Therefore, from a previous case, $A \sqsubseteq_\varphi \lambda z.Bz$ and, by definition, $\lambda z.Bz \sqsubseteq_\varphi B$.
- $A \equiv xA_1 \dots A_m$. Then $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta' \uplus \{x{:}\alpha_1 \to \dots \to \alpha_m \to \psi^{(n)}_i\}; \psi^{(n)}_i \rangle$, where $\mathsf{pp}^{(n)}_\varphi(A_j) = \langle \Delta_j; \alpha_j \rangle$, for $1 \le j \le m$ and $\Delta' = \biguplus_{j=1}^m \Delta_j$. So, by Lemma 48(1), $B \equiv xB_1 \dots B_m$ and $\Gamma \vdash_\varphi B_j : \alpha_j$, for $1 \le j \le m$. This implies, by induction, $A_j \sqsubseteq_\varphi B_j$, for $1 \le j \le m$, so we conclude $A \sqsubseteq_\varphi B$. $\square$

Lemma 53 *Assume $\varphi \in \{\mathtt{e}, \mathtt{i}\}$, $A, B \in \mathcal{A}_\varphi$. Let h be the height of $\mathcal{T}_\varphi(A)$, and n be such that $A, (B)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$. Then $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Gamma; \alpha \rangle$ and $\Gamma \vdash_\varphi B : \alpha$ imply $A \sqsubseteq_\varphi B$.*

PROOF. We prove by induction on A a stronger claim:

Assume $A, B \in \mathcal{A}_\varphi$. Let h be such that $(A)^{(h)}_\varphi \equiv A$, and n be such that $A, (B)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$. Moreover, let $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta; \alpha \rangle$ and Γ be a generalized special basis of degree n that contains Δ. Then $\Gamma \vdash_\varphi B : \alpha$ implies $A \sqsubseteq_\varphi B$.

The desired property is then obtained by taking $\Gamma = \Delta$.

- $A \equiv \lambda x.A'$. Then $\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta; \beta \to \alpha \rangle$, and, if $\beta \neq_\varphi \omega$, then $\langle \Delta, x{:}\beta; \alpha \rangle = \mathsf{pp}(A')$, and $\langle \Delta; \alpha \rangle = \mathsf{pp}(A')$ otherwise. If $B \equiv \lambda x.B'$, then $\Gamma \vdash_\varphi B : \beta \to \alpha$ implies $\Gamma, x{:}\beta \vdash_\varphi B' : \alpha$ and we are done by induction. If $B \equiv yB_1 \dots B_m$, then $\Gamma \vdash_\varphi B : \beta \to \alpha$ implies $\Gamma \vdash_\varphi \lambda z.Bz : \beta \to \alpha$, where $z \notin FV(B)$. Notice that, by Definitions 4 and 5, $\mathcal{T}_\varphi(\lambda z.Bz) = \mathcal{T}_\varphi(B)$, and, therefore, $(\lambda z.Bz)^{(h)}_\varphi = (B)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$. From the previous case we get $A \sqsubseteq_\varphi \lambda z.Bz$ and, by definition, $\lambda z.Bz \sqsubseteq_\varphi B$.
- $A \equiv xA_1 \dots A_m$. Then

$$\mathsf{pp}^{(n)}_\varphi(A) = \langle \Delta' \uplus \{x{:}\alpha_1 \to \dots \to \alpha_m \to \zeta^n \to \chi^{(n)}_i\}; \zeta^n \to \chi^{(n)}_i \rangle,$$

where $\mathsf{pp}^{(n)}_\varphi(A_j) = \langle \Delta_j; \alpha_j \rangle$ ($1 \le j \le m$), and $\Delta' = \biguplus_{j=1}^m \Delta_j$. Then, by Lemma 48(2), $B \equiv \lambda y_1 \ldots y_k.x B_1 \ldots B_m C_1 \ldots C_k$, $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi B_j : \alpha_j$, for $1 \le j \le m$, and $\Gamma \uplus \{y_1{:}\zeta, \ldots, y_k{:}\zeta\} \vdash_\varphi C_j : \zeta$, for $1 \le j \le k$. This implies, by induction, $A_j \sqsubseteq_\varphi B_j$, for $1 \le j \le m$, so we conclude, by definition, $A \sqsubseteq_\varphi B$. □

Theorem 54 (Main Theorem) *The following conditions are equivalent:*

(1) $\mathcal{T}_\varphi(M) = \mathcal{T}_\varphi(N)$*;*
(2) $\Gamma \vdash_\varphi M : \alpha$ *if and only if* $\Gamma \vdash_\varphi N : \alpha$*, for all* Γ, α*.*

PROOF.

(1) ⇒ (2) If M and N have the same trees, then they have the same sets of approximate normal forms, and, therefore, the same types by the Approximation Theorem (Theorem 43).

(2) ⇒ (1) If $\mathcal{T}_\varphi(M) \neq \mathcal{T}_\varphi(N)$, then by Lemma 18 we can find an $A \in \mathcal{A}_\varphi(M)$ such that there is no $B \in \mathcal{A}_\varphi(N)$ such that $A \sqsubseteq_\varphi B$ (or vice versa).

- For $\varphi = \mathtt{t}$, $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} M : \mathsf{ct}_{\mathtt{t}}(A)$ and $\Gamma_{\mathtt{t}} \not\vdash_{\mathtt{t}} N : \mathsf{ct}_{\mathtt{t}}(A)$, by the Approximation Theorem (Theorem 43) and Lemma 51(4).
- For $\varphi \in \{\mathtt{w}, \mathtt{h}\}$, let n be so big that $A \in \mathcal{A}^{(n)}_\varphi$ and $\langle \Gamma; \alpha \rangle = \mathsf{pp}^{(n)}_\varphi(A)$. We have by the Approximation Theorem (Theorem 43) and Lemma 52 that $\Gamma \vdash_\varphi M : \alpha$ and $\Gamma \not\vdash_\varphi N : \alpha$.
- For $\varphi \in \{\mathtt{e}, \mathtt{i}\}$, let h be the height of $\mathcal{T}_\varphi(A)$ and n be so big that $A, (N)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$. This implies $(B)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$, for all $B \in \mathcal{A}(N)$. Moreover, let $\langle \Gamma; \alpha \rangle = \mathsf{pp}^{(n)}_\varphi(A)$. Then we have, by the Approximation Theorem (Theorem 43) and Lemma 53, $\Gamma \vdash_\varphi M : \alpha$ and $\Gamma \not\vdash_\varphi N : \alpha$. □

In all cases, we get a discrimination algorithm, i.e., for two arbitrary terms M, N with different φ-trees, we can always find Γ and α such that $\Gamma \vdash_\varphi M : \alpha$ and $\Gamma \not\vdash_\varphi N : \alpha$, or vice versa. The least easy case is that of $\varphi \in \{\mathtt{e}, \mathtt{i}\}$. In this case we take an approximate normal form A such that $A \in \mathcal{A}_\varphi(M)$ and there is no $B \in \mathcal{A}_\varphi(N)$ such that $A \sqsubseteq_\varphi B$ (or vice versa). Let h be the height of $\mathcal{T}_\varphi(A)$ and n be so big that $A, (N)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$. This implies $(B)^{(h)}_\varphi \in \mathcal{A}^{(n)}_\varphi$, for all $B \in \mathcal{A}(N)$. Now we can choose $\langle \Gamma; \alpha \rangle = \mathsf{pp}^{(n)}_\varphi(A)$.

Example 55
- $\bot\bot \in \mathcal{A}_{\mathtt{t}}(\Omega_3)$ *and* $\mathcal{A}_{\mathtt{t}}(\lambda z.\Omega_2) = \{\bot, \lambda z.\bot\}$*. Then* $\mathsf{ct}_{\mathtt{t}}(\bot\bot) = \omega\omega$*, and* $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} \Omega_3 : \omega\omega$*, while* $\Gamma_{\mathtt{t}} \not\vdash_{\mathtt{t}} \lambda z.\Omega_2 : \omega\omega$*.*
- $\lambda z.\bot \in \mathcal{A}_{\mathtt{t}}(\lambda z.\Omega_2)$ *and, for all* $A \in \mathcal{A}_{\mathtt{t}}(\Omega_3)$*,* $\lambda z.\bot \not\sqsubseteq_\varphi A$*. We observe that* $\mathsf{ct}_{\mathtt{t}}(\lambda z.\bot) = \omega{\to}\omega$ *and* $\Gamma_{\mathtt{t}} \vdash_{\mathtt{t}} \lambda z.\Omega_2 : \omega{\to}\omega$*, while* $\Gamma_{\mathtt{t}} \not\vdash_{\mathtt{t}} \Omega_3 : \omega{\to}\omega$*.*
- $\lambda z.\bot \in \mathcal{A}_{\mathtt{w}}(\lambda z.\Omega_2)$ *and* $\mathcal{A}_{\mathtt{w}}(\Omega_3) = \{\bot\}$*. We get* $\mathsf{pp}^{(3)}_{\mathtt{w}}(\lambda z.\bot) = \langle \emptyset; \omega{\to}\omega \rangle$ *and* $\vdash_{\mathtt{w}} \lambda z.\Omega_2 : \omega{\to}\omega$*, while* $\not\vdash_{\mathtt{w}} \Omega_3 : \omega{\to}\omega$*.*

- $\Delta_2 \in \mathcal{A}_{\mathtt{h}}(\Delta_2)$ *and, for all* $A \in \mathcal{A}_{\mathtt{h}}(\Delta_2^\eta)$, $\Delta_2 \not\sqsubseteq_{\mathtt{h}} A$. *We get*

$$\begin{aligned}\mathsf{pp}^{(4)}_{\mathtt{w}}(\Delta_2) &= \langle \emptyset; \psi_1^{(4)} \wedge (\psi_1^{(4)} \to \psi_2^{(4)}) \to \psi_2^{(4)} \rangle,\\ \vdash_{\mathtt{h}} \Delta_2 &: \psi_1^{(4)} \wedge (\psi_1^{(4)} \to \psi_2^{(4)}) \to \psi_2^{(4)}, \textit{ and}\\ \not\vdash_{\mathtt{h}} \Delta_2^\eta &: \psi_1^{(4)} \wedge (\psi_1^{(4)} \to \psi_2^{(4)}) \to \psi_2^{(4)}.\end{aligned}$$

- $\mathbf{I} \in \mathcal{A}_{\mathtt{e}}(\mathbf{I})$ *and, for all* $A \in \mathcal{A}_{\mathtt{e}}(\mathbf{RR})$, $\mathbf{I} \not\sqsubseteq_{\mathtt{e}} A$. *We get*

$$\begin{aligned}\mathsf{pp}^{(3)}_{\mathtt{e}}(\mathbf{I}) &= \langle \emptyset; (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)} \rangle,\\ \vdash_{\mathtt{e}} \mathbf{I} &: (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)}, \textit{ and}\\ \not\vdash_{\mathtt{e}} \mathbf{RR} &: (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)}.\end{aligned}$$

- $\mathbf{I} \in \mathcal{A}_{\mathtt{i}}(\mathbf{RR})$ *and, for all* $A \in \mathcal{A}_{\mathtt{i}}(\mathbf{Y})$, $\mathbf{I} \not\sqsubseteq_{\mathtt{i}} A$. *We get*

$$\begin{aligned}\mathsf{pp}^{(3)}_{\mathtt{i}}(\mathbf{I}) &= \langle \emptyset; (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)} \rangle,\\ \vdash_{\mathtt{i}} \mathbf{RR} &: (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)}, \textit{ and}\\ \not\vdash_{\mathtt{i}} \mathbf{Y} &: (\zeta^{(3)} \to \chi_1^{(3)}) \to \zeta^{(3)} \to \chi_1^{(3)}.\end{aligned}$$

Acknowledgments

The authors are grateful to the anonymous referees for their helpful comments. The second author wishes to thank also Antonio Tinelli and Federica Lotti.

References

[1] Samson Abramsky and C.-H. Luke Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.

[2] Steffen van Bakel. Principal type schemes for the strict type assignment system. *Journal of Logic and Computation*, 3(6):643–670, 1993.

[3] Franco Barbanera, Mariangiola Dezani-Ciancaglini, and Fer-Jan de Vries. Types for trees. In *PROCOMET'98*, pages 6–29. Chapman & Hall, London, 1998.

[4] Henk Barendregt. *The lambda calculus. Its syntax and semantics*. North-Holland, Amsterdam, revised edition, 1984.

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

[6] Alessandro Berarducci. Infinite λ-calculus and non-sensible models. In *Logic and algebra*, pages 339–377. Dekker, New York, 1996.

[7] Alessandro Berarducci and Mariangiola Dezani-Ciancaglini. Infinite λ-calculus and types. *Theoretical Computer Science*, 212(1-2):29–75, 1999.

[8] Gérard Boudol. Lambda-calculi for (strict) parallel functions. *Information and Computation*, 108(1):51–127, 1994.

[9] Gérard Boudol and Cosimo Laneve. The discriminating power of multiplicities in the λ-calculus. *Information and Computation*, 126(1):83–102, 1996.

[10] Mario Coppo, Mariangiola Dezani-Ciancaglini, Furio Honsell, and Giuseppe Longo. Extended type structures and filter lambda models. In *Logic Colloquium '82*, pages 241–262. North-Holland, Amsterdam, 1984.

[11] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. Principal type schemes and λ-calculus semantics. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 535–560. Academic Press, London, 1980.

[12] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Maddalena Zacchi. Type theories, normal forms, and D_∞-lambda-models. *Information and Computation*, 72(2):85–116, 1987.

[13] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, and Adolfo Piperno. A filter model for concurrent λ-calculus. *SIAM Journal of Computing*, 27(5):1376–1419, 1998.

[14] Mariangiola Dezani-Ciancaglini, Benedetto Intrigila, and Marisa Venturini-Zilli. Böhm's theorem for Böhm trees. In *ICTCS'98*, pages 1–23. World Scientific, Oxford, 1998.

[15] Mariangiola Dezani-Ciancaglini, Paula Severi, and Fer-Jan de Vries. Böhm's theorem for Berarducci trees. In *CATS'00*, Electronic Notes in Theoretical Computer Science, Vol. 31(1), pages 143–166. Elsevier, Amsterdam, 2000.

[16] Mariangiola Dezani-Ciancaglini, Jerzy Tiuryn, and Paweł Urzyczyn. Discrimination by parallel observers: the algorithm. *Information and Computation*, 150(2):153–186, 1999.

[17] Erwin Engeler. Algebras and combinators. *Algebra Universalis*, 13(3):389–392, 1981.

[18] Martin Hyland. A syntactic characterization of the equality in some models for the lambda calculus. *Journal of the London Mathematical Society (2)*, 12(3):361–370, 1975/76.

[19] Richard Kennaway, Jan Willem Klop, Ronan Sleep, and Fer-Jan de Vries. Infinitary lambda calculus. *Theoretical Computer Science*, 175(1):93–125, 1997.

[20] Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries. Meaningless terms in rewriting. *Journal of Functional and Logic Programming*, Article 1, 35 pp. (electronic), 1999.

[21] Jean-Jacques Lévy. An algebraic interpretation of the $\lambda\beta K$-calculus, and an application of a labeled λ-calculus. *Theoretical Computer Science*, 2(1):97–114, 1976.

[22] Giuseppe Longo. Set-theoretical models of λ-calculus: theories, expansions, isomorphisms. *Annals of Pure and Applied Logic*, 24(2):153–188, 1983.

[23] Reiji Nakajima. Infinite normal forms for the λ-calculus. In *λ-calculus and computer science theory*, Lecture Notes in Computer Science, Vol. 37, pages 62–82. Springer Verlag, Berlin, 1975.

[24] Simonetta Ronchi della Rocca. Characterization theorems for a filter lambda model. *Information and Control*, 54(3):201–216, 1982.

[25] Davide Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Information and Computation*, 111(1):120–153, 1994.

[26] Dana Scott. Continuous lattices. In *Toposes, algebraic geometry and logic*, pages 97–136. Lecture Notes in Mathematics., Vol. 274. Springer Verlag, Berlin, 1972.

[27] Dana Scott. Data types as lattices. *SIAM Journal of Computing*, 5(3):522–587, 1976.

[28] Dana Scott. Domains for denotational semantics. In *Automata, languages and programming*, Lecture Notes in Computer Science, Vol. 140, pages 577–613. Springer Verlag, Berlin, 1982.

[29] Christopher P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus. *SIAM Journal of Computing*, 5(3):488–521, 1976.

Appendix

Definition 56 (Trees) *(1) A* tree *is a prefix-closed set of sequences of non-zero natural numbers such that, if $s.(n+1)$ belongs to a tree, then so does $s.n$.*

(2) A labeled tree *is a tree T equipped with a label function $\mathcal{L}_T : T \to L$, where L is the set of labels.*

The set formed only by the empty sequence will denote the one-node tree, whereas the empty tree is denoted by the empty set. In the above definition, a node is identified with the path connecting it to the root.

We denote by $|s|$ the length of the sequence s. If $|s| = k$ and $h \leq k$, we can define $s_{|h}$ as the prefix of s of length h.

Definition 57 (Tree pruning) *(1) Given a tree T, we define $T_{|n}$,* the pruning of T at level n*, as the set of sequences in T of length $\leq n$, i.e.:*

$$T_{|n} = \{s \in T \mid |s| \leq n\}.$$

Let T be a labeled tree. The label function $\mathcal{L}_{T_{|n}}$ of $T_{|n}$ is the obvious restriction of the label function of T.

(2) *Given a labeled tree T and a function $f : L \to L$, where L is the set of labels of T, we define $T_{|n,f}$ as follows:*
- *the set of nodes of $T_{|n,f}$ coincides with that of $T_{|n}$.*
- *the label function $\mathcal{L}_{T_{|n,f}}$ is defined by $\mathcal{L}_{T_{|n,f}}(s) = \mathcal{L}_{T_{|n}}(s)$ if $|s| \leq n-1$, $\mathcal{L}_{T_{|n,f}}(s) = f(s)$ otherwise.*

Let $L_{\mathtt{t}} = \{\perp, @, x, \lambda x \mid x \text{ is a variable }\}$, $L_{\mathtt{w}} = \{\perp, x, \lambda x \mid x \text{ is a variable }\}$, and $L_{\mathtt{h}} = L_{\mathtt{e}} = L_{\mathtt{i}} = \{\perp, \lambda y_1 \ldots y_n.x \mid y_1, \ldots, y_n, x \text{ are variables and } n \geq 0\}$. Then the φ-trees are labeled trees with sets of labels L_φ.

Definition 58 (Infinite η-expansion) *Let T and T' be two head-trees and define $f(\lambda y_1 \ldots y_n.x) = x$ $(n \geq 0)$, $f(\perp) = \perp$. Then* [6]

$$T \geq_\eta T' \Leftrightarrow \forall n.\eta(T_{|n,f}) = T'_{|n,f}.$$

Given a finite φ-tree T, it is easy to find the approximate normal form $A \in \mathcal{A}_\varphi$ such that $\mathcal{T}_\varphi(A)$ is T. For example, in the case of top trees we have the following mapping $\mathsf{m}_{\mathtt{t}} : \mathcal{T}_{\mathtt{t}} \to \mathcal{A}_{\mathtt{t}}$:

$$\mathsf{m}_{\mathtt{t}}\left(\begin{array}{ccc} & x & \\ \diagup & & \diagdown \\ T_1 & \ldots & T_m \end{array}\right) = x\,\mathsf{m}_{\mathtt{t}}(T_1)\ldots\mathsf{m}_{\mathtt{t}}(T_m) \qquad \mathsf{m}_{\mathtt{t}}\left(\begin{array}{c} \lambda x \\ | \\ T \end{array}\right) = \lambda x.\mathsf{m}_{\mathtt{t}}(T)$$

$$\mathsf{m}_{\mathtt{t}}\left(\begin{array}{ccc} & @ & \\ \diagup & & \diagdown \\ T & & T' \end{array}\right) = \mathsf{m}_{\mathtt{t}}(T)\mathsf{m}_{\mathtt{t}}(T') \qquad \mathsf{m}_{\mathtt{t}}(\perp) = \perp.$$

The definitions of $\mathsf{m}_\varphi : \mathcal{A}_\varphi \to \mathcal{T}_\varphi$ for $\varphi \neq \mathtt{t}$ are similar.

Definition 59 (Cut with $\perp$) *If g is the constant function always returning $\perp$, then $(M)^{(h)}_\varphi$, i.e., the approximate normal form whose tree is $\mathcal{T}_\varphi(M)_{|h,g}$, is defined by*

$$(M)^{(h)}_\varphi = \mathsf{m}_\varphi(\mathcal{T}_\varphi(M)_{|h,g}).$$

[6] The present definition of $\geq_\eta$ differs from the original ([4], Definition 10.2.10), but they coincide when T' is a single node whose label is a variable, and this is the only case used in the definition of infinite eta trees (Definition 5).