6

Types for Trees

F. Barbanera, M. Dezani-Ciancaglini
Università degli studi di Torino
Dipartimento di Informatica, C.so Svizzera 185, 10149 Torino, Italia
Email: barba, dezani@di.unito.it

F.J. de Vries
Computer Science Division, ETL
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 terms. More precisely, for each family of trees two terms have the same tree if and only if they get assigned the same types in the corresponding type assignment system.

Keywords

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 the meaning of a λ-term (program) is represented by the amount of "meaningful information" we can extract from the term by "running it". The formalization of "the information" obtained from a term requires first the definition of what is, in a λ-term, a "stable relevant minimal information" directly observable in the 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 the $\sqrt{2}$. The calculation process merely adds decimals to the already calculated decimal expansion).

All stable relevant minimal information produced during a computation can quite naturally be represented as a tree. This tree then embodies the total information hidden in the original term. There are many such a tree representation in literature, depending on the possible notions of stable relevant minimal information; the

Programming Concepts and Methods D. Gries, & W-P. de Roever (Eds.)

most commonly used being *top* trees (or Berarducci trees (Berarducci 1996)), *weak* trees (or Lévy-Longo trees (Sangiorgi 1994)), *head* trees (or Böhm trees (Barendregt 1984)), *eta* trees and *infinite eta* trees (infinite eta trees are in one-one correspondence with Nakajima trees (Nakajima 1975)). Hence the various notions of tree represent different notions of *meaning* of a term (in particular they also specify different notions of *undefined* value).
This apparently vague intuition is substantiated by results starting with (Wadsworth 1976), which show that there exist precise correspondences between the tree representations of terms and the local structures (or equivalently the λ-theories) of certain λ-models ((Barendregt 1984), Chapter 19). In particular, such correspondences amount to the fact that two λ-terms have the same tree representation iff 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 (Scott 1972) (this result is proved in (Wadsworth 1976));
- the eta trees represent the local structure of the inverse limit model defined in (Coppo et al. 1987);
- the head trees represent the local structure of Scott's P_ω model as defined in (Scott 1976) (a discussion on this topic can be found in (Barendregt 1984), Chapter 19);
- the weak trees were introduced by Longo in (Longo 1983) (following (Lévy 1976)), who proved that they represent the local structure of Engeler models as defined in (Engeler 1981).

Orthogonally, the results about observational equivalences confirm this operational intuition of *dynamically* evolving meanings of terms encorporated in the tree representations. For instance in (Wadsworth 1976) Wadsworth shows that two λ-terms M, N have the same infinite Böhm tree iff for all contexts $C[\,]$ the following holds:

$$C[M] \text{ has a head normal form } \Leftrightarrow \; C[N] \text{ has a head normal form.}$$

The same property holds even considering eta trees and normal forms (Hyland 1976). Weak trees correspond, instead, to the observational equivalence with respect to weak head normal forms in suitably enriched versions of the λ-calculus (Sangiorgi 1994, Boudol et al. 1996, Dezani et al. 1997).

λ-terms represent programs which themselves are *static* entities and as such can be "handled", in contrast to the more ineffable computations. It would be very useful if these dynamic aspects could be analysed with tools dealing with static entities like terms and types. 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 terms.

- Each inverse limit λ-model is isomorphic to a filter model, i.e. to a model in which the meanings of terms are sets of derivable intersection types (Coppo et al. 1983).

- Two terms have the same Böhm tree iff they have the same set of types in the standard intersection type discipline (Barendregt et al. 1983), as proved in (Ronchi della Rocca 1982).
- Two terms have the same Lévy-Longo tree iff they have the same set of types in the type discipline with union and intersection of (Dezani et al. to appear), as proved in (Dezani et al. 1997).
- Two terms have the same Berarducci trees iff they have the same set of types in a type assignment system with applicative types (Berarducci et al. to appear).

In the present paper we 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 show that two 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 (Ronchi della Rocca 1982) since we do not consider infinite type variables;
- we do not allow the union type constructor which is considered instead in (Dezani et al. 1997);
- the applicative types are build starting from just one constant instead of two (this last is the choice of (Berarducci et al. to appear)).

All the type systems we introduce (but that representing Beraducci trees) induce filter λ-models in the sense of (Barendregt et al. 1983). Clearly the theories of these filter models coincide with the equalities of the corresponding trees. So as byproduct we obtain alternative proofs of the characterizations of the theories of Scott's D_∞ model (Wadsworth 1976) and of the filter λ-model (Ronchi della Rocca 1982). Notice that these new proofs (unlike the original ones) are constructive, in the sense that, whenever two terms have different interpretations, we build a compact d such that d approximates only the interpretation of one of the two terms. Really d is the principal filter induced by a type which can be deduced only for one of the two 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 terms having different trees?", which can be formulated for each family of trees.
An intersection type assignment system played a crucial role to show that observational equivalence in lazy concurrent λ-calculus is equivalent to Lévy-Longo tree equality (Dezani et al. 1997). 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 us to search for a proof along the following lines.
Suppose we were able to define for each type α a test term T_α such that $T_\alpha M$ converges iff M has type α. Then we would obtain an observational equivalence which coincides with the tree equality (see (Boudol 1994)).

The paper is organized as follows. In Section 2 we shall recall the various definitions of tree. Moreover, we shall introduce the notion of *approximant*. In Section 3 we shall describe the type assigment systems which will be used for our main result and we shall give a theorem of approximation stating that a term has a type iff there exists an approximant of the term which has the same type. Section 4, instead, contains our main result.
Syntax, basic notation on the λ-calculus and the usual conventions on variables to prevent explicit mentioning of α-conversion are as in (Barendregt 1984).

The proofs we omitted for reasons of space can be found in the full version of the paper which can be obtained from the URL:
ftp://lambda.di.unito.it/pub/dezani/tm.ps.gz.

2 TREES AND APPROXIMANTS

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* (Kennaway et al. 1996)).
If during a computation we get the following terms, their underlined parts will remain stable during the rest (if any) of the computation: $\underline{x}M_1 \dots M_m$, $\underline{\lambda x.M}$, $\underline{P@Q}$ (where @ is the explicit representation of the operation of application and P is a 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 $\lambda y_1 \dots y_n.zN_1 \dots N_m$ $(n, m \geq 0)$. This means that we can end up with different notions of what a *stable relevant minimal information* is.
In order to formalize such notions it is possible to define for each notion a reduction relation such that

(a) if a term can produce a stable relevant minimal information we can get it by means of the given reduction relation;

(b) the computation process represented by the reduction relation stops once a stable relevant minimal information is obtained.

In the following we give a number of such reduction relations for λ-terms present in the literature. All of them are proper restrictions of the usual β-reduction relation.

Definition 1 *Given the following axioms and rules:*

(β) $\quad (\lambda x.M)N \to M[N/x]$
(η) $\quad \lambda x.Mx \to M \quad if\ x \notin FV(M)$
(ν) $\quad M \to N \ \Rightarrow \ ML \to NL$
$(\nu)_t$ $\quad M \to N \ \Rightarrow \ ML \to NL$ $\quad$ (*proviso* M *is not a* strong zero term*).

*That is, it cannot be reduced to a weak head normal form (as defined in Definition 2.2.) by means of the reduction relation induced by the (β) rule (Berarducci 1996). Such terms are called *unsolvables of order* 0 in (Longo 1983) and *strongly unsolvables* in (Abramsky et al. 1993).

$$(\xi) \qquad M \to N \;\Rightarrow\; \lambda x.M \to \lambda x.N$$

we can define the following reduction relations on λ-terms.

(top reduction) $\to_{\mathtt{t}}$ *is the relation relation induced by* (β) *and* $(\nu)_{\mathtt{t}}$
(weak head reduction) $\to_{\mathtt{w}}$ *is the reduction relation induced by* (β) *and* (ν)
(head reduction) $\to_{\mathtt{h}}$ *is the reduction relation induced by* (β), (ν) *and* (ξ)
(eta head reduction) $\to_{\mathtt{e}}$ *is the reduction relation induced by* (β), (ν), (ξ) *and* (η).

The weak head reduction is better known as lazy reduction (Abramsky et al. 1993).

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

Definition 2 *1. A* top normal form* *is a term of one of the following three kinds:*

(a) an application term of the form $xM_1 \dots M_m$ $(m \geq 0)$*;*
(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 $xM_1 \dots M_m$ $(m \geq 0)$*;*
(b) an abstraction term $\lambda x.M$*.*

3. A head normal form *is a term of the following kind:*

(a) $\lambda x_1 \dots x_n.yM_1 \dots M_m$ $(m, n \geq 0)$*.*

4. An eta head normal form *is a term of the following kind:*

(a) $\lambda x_1 \dots x_n.yM_1 \dots M_m$ $(m, n \geq 0)$,
where $x_n \not\equiv M_m$ *or* $x_n \in FV(yM_1 \dots M_{m-1})$.

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 normal forms, etc.

Example 3 *Let* $\mathbf{\Delta_n} \equiv \lambda x.\underbrace{x \dots x}_{n}$ *and* $\mathbf{\Omega_n} \equiv \underbrace{\mathbf{\Delta_n} \dots \mathbf{\Delta_n}}_{n}$*. Moreover, let* $Q \equiv \lambda x.f(xx)$, $R \equiv \lambda zxy.x(zzy)$ *and* $\mathbf{\Delta_2^\eta} \equiv \lambda xy.xxy$*. Then*

1. For each $n \geq 2$ *the term* $\mathbf{\Omega_n}$ *is an example of strong zero term.*
2. $\mathbf{\Omega_2}$ *is not a top normal form, while* $\mathbf{\Omega_n}$ *(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 which cannot reduce to head normal form.*
4. $\mathbf{\Delta_2^\eta}$ *is a head normal form but it is not an eta head normal form.*
5. $\lambda f.f(QQ)$ *and* $\lambda xy.x(RRy)$ *are eta head normal forms.*

---

*called *root stable form* in (Kennaway et al. 1997).

Following (Berarducci 1996) we say that a term is *mute* iff it does not have a top normal form.
With this definition we can represent in tree notation all the various related kinds of information we can distract from a term. Given a term M for each of the four reduction relations we can try to reduce M to normal form. If it has no normal form then no information is obtainable out of M and its tree is $\bot$. Otherwise, put the information so obtained in a node and build the children of such a 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 (*i*) *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:*

$$\mathcal{T}_{\mathtt{t}}(M) = \begin{array}{ccccc} & & x & & \\ & \swarrow & & \searrow & \\ \mathcal{T}_{\mathtt{t}}(N_1) & & \ldots & & \mathcal{T}_{\mathtt{t}}(N_m) \end{array}$$

- *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:*

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

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

(*ii*) *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:*

$$\mathcal{T}_{\mathtt{w}}(M) = \begin{array}{ccccc} & & x & & \\ & \swarrow & & \searrow & \\ \mathcal{T}_{\mathtt{w}}(N_1) & & \ldots & & \mathcal{T}_{\mathtt{w}}(N_m) \end{array}$$

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

$$\mathcal{T}_{\mathrm{w}}(M) = \quad \lambda x \;\Big|\; \mathcal{T}_{\mathrm{w}}(N)$$

– *otherwise:* $\mathcal{T}_{\mathrm{w}}(M) = \bot$

(*iii*) *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:*

$$\mathcal{T}_{\mathrm{h}}(M) = \quad \lambda x_1 \ldots x_n.y \;\diagup\diagdown\; \mathcal{T}_{\mathrm{h}}(N_1) \quad \ldots \quad \mathcal{T}_{\mathrm{h}}(N_m)$$

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

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

– *if* $M \to_{\mathrm{e}} \lambda x_1 \ldots x_n.yN_1 \ldots N_m$ $(n, m \geq 0)$, *where* $x_n \not\equiv N_m$ *or* $x_n \in FV(yN_1 \ldots N_{m-1})$, *then:*

$$\mathcal{T}_{\mathrm{e}}(M) = \quad \lambda x_1 \ldots x_n.y \;\diagup\diagdown\; \mathcal{T}_{\mathrm{e}}(N_1) \quad \ldots \quad \mathcal{T}_{\mathrm{e}}(N_m)$$

– *otherwise:* $\mathcal{T}_{\mathrm{e}}(M) = \bot$

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 (Barendregt 1984). 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 one:

$$x \;{}_{\eta}\!\leftarrow \mathcal{T}_{\mathrm{h}}(\lambda z_0.xz_0) \;{}_{\eta}\!\leftarrow \mathcal{T}_{\mathrm{h}}(\lambda z_0.x(\lambda z_1.z_0z_1)) \;{}_{\eta}\!\leftarrow \ldots$$

We denote by $T \geq_{\eta} x$ the fact that T is a (possibly) infinite η-expansion of x. We refer to (Barendregt 1984) (10.2.10) for a formal definition of $\geq_{\eta}$.

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, $\infty\eta(T)$, is defined as follows:*

$$\infty\eta(\bot) = \bot$$

$$\infty\eta\left(\begin{array}{c}\lambda x_1 \ldots x_n.y \\ /\backslash \\ T_1 \ldots T_m\end{array}\right) = \begin{cases} \infty\eta\left(\begin{array}{c}\lambda x_1 \ldots x_{n-1}.y \\ /\backslash \\ T_1 \ldots T_{m-1}\end{array}\right) & \text{if } x_n \leq_\eta T_m \text{ and } x_n \notin FV(T_i),\ 1 \leq i \leq m-1 \\ \begin{array}{c}\lambda x_1 \ldots x_n.y \\ /\backslash \\ \infty\eta(T_1) \ldots \infty\eta(T_m)\end{array} & \text{otherwise} \end{cases}$$

It is possible now to define *infinite eta trees.*

Definition 6 *The* infinite eta tree $\mathcal{T}_{\mathtt{i}}(M)$ *of a term M is defined as* $\infty\eta(\mathcal{T}_{\mathtt{e}}(M))$.

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

As usual when dealing with (possibly) infinite structures, one can consider finite approximations. In our case, we can get approximations by pruning our trees. We need to add a constant in our language ($\bot$) in order to represent the (possibly infinite) parts of the trees that have been pruned. An approximation of a tree then represents certain finite stable information that can be extracted from a term.

Given a notion of tree, approximations of a tree can be represented as terms in a slightly enriched λ-calculus.

Let $\Lambda_\bot$ be the set of terms obtained by adding the symbol $\bot$ to the syntax of the pure λ-calculus.

We extend the notion of $\to_\dagger$-reduction ($\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}\}$) to $\Lambda_\bot$ by adding the clauses

$$\bot M \to \bot \quad \text{for } \dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}\}, \quad \lambda x.\bot \to \bot \quad \text{for } \dagger \in \{\mathtt{h}, \mathtt{e}\}.$$

Definition 7 *For $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}\}$, we define $\mathcal{A}_\dagger \subseteq \Lambda_\bot$ as the set of normal forms with respect to $\to_\dagger$. $\mathcal{A}_{\mathtt{i}}$ is defined to be equal to $\mathcal{A}_{\mathtt{e}}$.*

We denote by $\mathcal{A}_\dagger^{(n)}$ ($\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$) the set of approximate normal forms with at most n symbols.

$(M)_\dagger^{(h)}$ ($\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$) denotes the approximate normal form representing the tree obtained out of $\mathcal{T}_\dagger(M)$ by cutting it at height h and putting as leaves at the end of the cut edges the constant $\bot$.

It is trivial to verify that $(M)_\dagger^{(h)} \in \mathcal{A}_\dagger$ for all M. Since the tree representations easily extend to approximate normal forms, we define $(A)_\dagger^{(h)}$ similarly. For instance,

by looking at $\mathcal{T}_{\mathtt{t}}(\mathbf{\Delta_3\Delta_3})$, it is easy to verify that $(\mathbf{\Delta_3\Delta_3})_{\mathtt{t}}^{(h)}$ for $h = 0, 1, 2, 3$ are $\bot$, $\bot\bot$, $\bot\bot(\lambda x.\bot)$ and $\bot\bot(\lambda x.\bot)(\lambda x.x\bot\bot)$.

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

Definition 8 *For* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, *the relation* $\preceq_\dagger$ *is the least partial order on* $\mathcal{A}_\dagger$, *such that:*

(*a*) $\bot \preceq_\dagger A$;
(*b*) *if* $A \preceq_\dagger A'$, *then* $\lambda x.A \preceq_\dagger \lambda x.A'$;
(*c*) *if* $A_i \preceq_\dagger A_i'$ *for* $i = 1, \ldots, n$, *then* $A_1 \ldots A_n \preceq_\dagger A_1' \ldots A_n'$.

It is trivial to verify that $(M)_\dagger^{(h)} \preceq_\dagger (M)_\dagger^{(h+1)}$ for all h.

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

Definition 9 *For* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$ *the* set $\mathcal{A}_\dagger(M)$ of approximants of M *is defined by:*

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

For example $\mathcal{A}_{\mathtt{t}}(x(\mathbf{\Omega_3 I})(\mathbf{II}))$ contains $\bot \preceq_{\mathtt{t}} x\bot\bot \preceq_{\mathtt{t}} x(\bot\mathbf{I})\mathbf{I} \preceq_{\mathtt{t}} x(\bot\mathbf{\Delta_3 I})\mathbf{I} \preceq_{\mathtt{t}} x(\bot\mathbf{\Delta_3\Delta_3 I})\mathbf{I} \preceq_{\mathtt{t}} \ldots$.

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

Theorem 10 *For* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$ *and for any* M, N:

$$\mathcal{T}_\dagger(M) = \mathcal{T}_\dagger(N) \quad \textit{iff} \quad \mathcal{A}_\dagger(M) = \mathcal{A}_\dagger(N).$$

One can show also that $\mathcal{T}_\dagger(M)$ is the least upper-bound of $\mathcal{A}_\dagger(M)$ wrt $\preceq_\dagger$.

It is possible to extend each partial order $\preceq_\dagger$ to a partial order $\sqsubseteq_\dagger$ naturally inducing an equivalence relation on sets of approximants. Such equivalence can be proved to coincide with the identity relation on sets of approximants and hence (by Theorem 10) to coincide with the identity on trees.

Definition 11 *1. Let* $\sqsubseteq_{\mathtt{t}}$ *be the relation* $\preceq_{\mathtt{t}}$.
2. *For* $\dagger \in \{\mathtt{w}, \mathtt{h}\}$ *the relation* $\sqsubseteq_\dagger$ *is the least partial order on* $\mathcal{A}_\dagger$ *which satifies clauses (a),(b),(c) of* $\preceq_\dagger$ *and, moreover,*

(*d*) $\lambda y.xA_1 \ldots A_n y \sqsubseteq_\dagger xA_1 \ldots A_n$ *for all variables* $y \notin FV(xA_1 \ldots A_n)$ $(1 \leq i \leq n)$.

3. *For* $\dagger \in \{\mathtt{e}, \mathtt{i}\}$ *the relation* $\sqsubseteq_\dagger$ *is the least partial order on* $\mathcal{A}_\dagger$ *which satifies clauses (a),(b),(c) of* $\preceq_\dagger$ *and, moreover,*
(*d*) *if* $A \sqsubseteq_\dagger xA_1 \ldots A_n y$ *where* $y \notin FV(xA_1 \ldots A_n)$, *then* $\lambda y.A \sqsubseteq_\dagger xA_1 \ldots A_n$;

(e) $xA_1 \dots A_n \sqsubseteq_\dagger \lambda z_1 \dots z_m . xA_1 \dots A_n B_1 \dots B_m$, *where* $x \notin \{z_1, \dots, z_m\}$ *and* $B_j \not\simeq \bot$ *for* $1 \leq j \leq m$.

Definition 12 *For* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$ *and any two terms* M *and* N *we define:* $\mathcal{A}_\dagger(M) \simeq_\dagger \mathcal{A}_\dagger(N) \Leftrightarrow$ *for all* $A \in \mathcal{A}_\dagger(M)$ *there is* $B \in \mathcal{A}_\dagger(N)$ *such that* $A \sqsubseteq_\dagger B$, *and vice versa.*

Lemma 13 *For* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$,

$$\mathcal{A}_\dagger(M) \simeq_\dagger \mathcal{A}_\dagger(N) \quad \textit{iff} \quad \mathcal{A}_\dagger(M) = \mathcal{A}_\dagger(N) \quad \textit{iff} \quad \mathcal{T}_\dagger(M) = \mathcal{T}_\dagger(N).$$

The main motivation for the introduction of $\sqsubseteq_\dagger$ is that it can be proved to be compatible with the typings that we shall introduce in the next section.

3 TYPES AND TYPE ASSIGNMENT SYSTEMS

As stated in the introduction, our static tools to analyse trees (or, equivalently, their corresponding sets of approximants) will be type assignment systems, in particular type assignment systems using 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 and α 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: $\mathbf{Types}_{\mathtt{t}}$ to characterize $\mathcal{T}_{\mathtt{t}}$, $\mathbf{Types}_{\mathtt{wh}}$ to characterize $\mathcal{T}_{\mathtt{w}}$ and $\mathcal{T}_{\mathtt{h}}$, and $\mathbf{Types}_{\mathtt{ei}}$ to characterize $\mathcal{T}_{\mathtt{e}}$ and $\mathcal{T}_{\mathtt{i}}$.
After defining these sets of types, in this section we shall define an order $\leq_\dagger$ 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 will be able to detect that a term carries no information: $\mathcal{T}_\dagger(M) = \bot$ if and only if ω is the only type that the type system related to $\mathcal{T}_\dagger$ can assign to M.
In the following, we shall use the following notation: if $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, then

$$\ddagger = \begin{cases} \mathtt{t} & \text{if } \dagger = \mathtt{t} \\ \mathtt{wh} & \text{if } \dagger = \mathtt{w} \text{ or } \dagger = \mathtt{h} \\ \mathtt{ei} & \text{if } \dagger = \mathtt{e} \text{ or } \dagger = \mathtt{i}. \end{cases}$$

3.1 Types

Let us begin with $\mathbf{Types}_{\mathtt{t}}$. To describe a top normal form which is the application of two terms, following (Berarducci et al. to appear) we introduce a particular type constructor: the application $\alpha\beta$ between 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 (Berarducci et al. to appear) since we build types starting only from the unique constant ω, i.e. we don't introduce a new type constant to be interpreted as the set of all strong zero terms.
We have to prevent the presence of inconsistent types. For example, $\omega\omega$ says that a top normal form is the application of two terms, the first one being a strong zero term, whereas $\omega \to \omega$ says that a top normal form is an abstraction. So we need to prevent their intersection $\omega\omega \wedge (\omega \to \omega)$.
To describe $\mathbf{Types}_t$ we can define first a set of "pretypes" and then restrict it.

In writing types we assume the following precedence between operators: application, intersection, arrow. Moreover, $\alpha \to \beta^n \to \gamma$ is short for $\alpha \to \underbrace{\beta \ldots \to \beta}_{n} \to \gamma$.

Definition 14 (Pretypes) *The set of pretypes* $\mathbf{PTypes}$ *is defined by the grammar:*

$$P ::= \omega \mid P \to P \mid PP \mid P \wedge P.$$

Definition 15 ($\mathbf{Types}_t$) *Given* $\alpha \in \mathbf{PTypes}$*, we define two predicates* $\alpha \in \mathbf{Types}_t$ *and* $\alpha \notin \mathbf{Types}_t$ *by simultaneous induction on* α *by stipulating that* $\alpha \in \mathbf{Types}_t$ *iff one of the following conditions holds (and* $\alpha \notin \mathbf{Types}_t$ *iff 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* $\alpha_i, \beta_i \in \mathbf{Types}_t$ *and for all* $J \subseteq I$ *either* $\bigwedge_{j\in J} \beta_j \in \mathbf{Types}_t$ *or* $\bigwedge_{j\in J} \alpha_j \notin \mathbf{Types}_t$.
(Applicative kind) α *is* $\omega\beta$*, or* α *is a finite intersection of the form* $\bigwedge_{i\in I} \alpha_i\beta_i$ *where* $\bigwedge_{i\in I} \beta_i \in \mathbf{Types}_t$*, and* $\bigwedge_{i\in I} \alpha_i \in \mathbf{Types}_t$ *is of applicative kind.*
If $\alpha \in \mathbf{Types}_t$*, then* $\omega \wedge \alpha \in \mathbf{Types}_t$*: the kind of* $\omega \wedge \alpha$ *is defined to be the kind of* α.

In what follows we shall consider only types. 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*).

To describe normal forms other than top normal forms, applicative types are not needed. Moreover, since all the intersections are meaningful, the definition of $\mathbf{Types}_{\mathtt{wh}}$ and $\mathbf{Types}_{\mathtt{ei}}$ 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 *. Roughly speaking, ζ can be seen as the collapse of all the applicative types.

Definition 16 ($\mathbf{Types}_{\mathtt{wh}}$) *The set of types* $\mathbf{Types}_{\mathtt{wh}}$ *is defined by the grammar:*

$$T_{\mathtt{wh}} ::= \omega \mid \zeta \mid T_{\mathtt{wh}} \to T_{\mathtt{wh}} \mid T_{\mathtt{wh}} \wedge T_{\mathtt{wh}}.$$

* Sangiorgi in (Sangiorgi 1994) proves that $\lambda x.xx$ 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.

In order to define $\mathbf{Types}_{\mathtt{ei}}$, since terms are considered modulo η, we are forced to equate all atomic types to intersections of arrow types. This means that another type constant, ϑ, is needed.

Definition 17 ($\mathbf{Types}_{\mathtt{ei}}$) *The set of types* $\mathbf{Types}_{\mathtt{ei}}$ *is defined by the grammar:*

$$T_{\mathtt{ei}} ::= \omega \mid \zeta \mid \vartheta \mid T_{\mathtt{ei}} \to T_{\mathtt{ei}} \mid T_{\mathtt{ei}} \wedge T_{\mathtt{ei}}.$$

3.2 Type preorders

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

The preorder $\leq_{\mathtt{t}}$ defined on $\mathbf{Types}_{\mathtt{t}}$ reflects the interpretation of applicative types. The preorder $\leq_{\mathtt{h}}$ defined on $\mathbf{Types}_{\mathtt{wh}}$ equates ω to $\omega \to \omega$, since we want to take into account the reduction $\lambda x.\bot \to \bot$. 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 argument of this arrow type is always ω, while this is not true for $\leq_{\mathtt{e}}$. This difference is essential to mimic either infinite or finite η-reductions, as we shall see later.

Definition 18 *1. Let* $\leq_{\mathtt{t}}$ *be the smallest binary relation over* $\mathbf{Types}_{\mathtt{t}}$ *such that:*

- *(a) it is a preorder in which* $\wedge$ *is the meet and* ω *is the top;*
- *the arrow satisfies:*

$$(b)\ \alpha \to \omega \leq \omega \to \omega; \quad (c)\ (\alpha \to \beta) \wedge (\alpha \to \gamma) \leq \alpha \to \beta \wedge \gamma;$$

$$(d)\ \alpha \geq \alpha' \text{ and } \beta \leq \beta' \text{ imply } \alpha \to \beta \leq \alpha' \to \beta'.$$

- *the applicative types satisfy:*

$$(e)\ \pi\alpha \wedge \pi'\alpha' \leq (\pi \wedge \pi')(\alpha \wedge \alpha'); \quad (f)\ \pi \leq \pi' \text{ and } \alpha \leq \alpha' \text{ imply } \pi\alpha \leq \pi'\alpha'.$$

2. *Let* $\leq_{\mathtt{w}}$ *be the smallest binary relation over* $\mathbf{Types}_{\mathtt{wh}}$ *which satisfies the clauses (a)-(d) above.*
3. *Let* $\leq_{\mathtt{h}}$ *be the smallest binary relation over* $\mathbf{Types}_{\mathtt{wh}}$ *which satisfies the clauses (a)-(d) above and moreover:*

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

4. *Let* $\leq_{\mathtt{e}}$ *be the smallest binary relation over* $\mathbf{Types}_{\mathtt{ei}}$ *which satisfies the clauses (a)-(d), (g) above and moreover:*

$$(h)\ \zeta \leq \vartheta \to \zeta \leq \zeta; \quad (i)\ \vartheta \leq \zeta \to \vartheta \leq \vartheta.$$

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

$$(l)\ \zeta \leq \omega \to \zeta \leq \zeta; \quad (m)\ \vartheta \leq \omega \to \vartheta \leq \vartheta.$$

$\alpha =_\dagger \beta$ is short for $\alpha \leq_\dagger \beta$ and $\beta \leq_\dagger \alpha$, where $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.

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

For example we have $\omega \wedge \alpha =_\dagger \alpha$ for every type $\alpha \in \mathbf{Types}_\ddagger$ where $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, $(\omega \to \omega) \wedge \sigma =_\mathtt{t} \sigma$ for every arrow type $\sigma \in \mathbf{Types}_\mathtt{t}$, $(\omega \to \omega) \wedge \alpha =_\dagger \alpha$ for every type $\alpha \in \mathbf{Types}_\ddagger$ (proviso that $\alpha \neq_\mathtt{w} \omega$ when $\dagger = \mathtt{w}$) where $\dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$, and $\omega\omega \wedge \pi = \pi$ for every applicative type $\pi \in \mathbf{Types}_\mathtt{t}$.

3.3 Type assignment systems

For each preorder introduced in the previous subsection, we define a type assignment system associating λ-terms to the corresponding set of types. 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 $\{\vdash_\dagger\}_{\dagger \in \{\mathtt{w},\mathtt{h},\mathtt{e},\mathtt{i}\}}$ are defined by such six rules, by instantiating rule $(\leq_\dagger)$ with the corresponding preorder. However, to define $\vdash_\mathtt{t}$ we have to deal with applicative types, and hence we need two extra rules: (ωapp) and (app). Moreover, a rule (Eq_β) is needed as well, since applicative types are not invariant under β-expansion of subjects. For example, without (Eq_β) we have $\vdash \Omega_2\mathbf{I}: \omega(\omega \to \omega)$, but $\not\vdash (\lambda xy.y\mathbf{\Delta}_2 x)\mathbf{I}\mathbf{\Delta}_2 : \omega(\omega \to \omega)$.

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 $\mathbf{Types}_\mathtt{t}$, $\mathbf{Types}_\mathtt{wh}$, and $\mathbf{Types}_\mathtt{ei}$, respectively.

Definition 19 (Type assignment systems) *Let us consider the following axioms and rules (where M, N are terms or approximate normal forms)*

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

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

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

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

$$(Eq_\beta)\ \frac{\Gamma \vdash N{:}\alpha \quad M =_\beta N}{\Gamma \vdash M{:}\alpha}.$$

1. The type assignment system $\vdash_\mathtt{t}$ is defined by the axioms and rules (Ax), (ω),

$(\to \mathrm{I})$, $(\to \mathrm{E})$, $(\wedge \mathrm{I})$, (ωapp), (app), (Eq_β), *and* $(\leq_\mathtt{t})$, *where* $\Gamma \in \mathcal{B}_\mathtt{t}$ *and* $\alpha, \beta \in$ $\mathbf{Types}_\mathtt{t}$.

2. *The type assignment system* $\vdash_\dagger$ *for* $\dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$ *is defined by the axioms and rules* (Ax), (ω), $(\to \mathrm{I})$, $(\to \mathrm{E})$, $(\wedge \mathrm{I})$, *and* $(\leq_\dagger)$, *where* $\Gamma \in \mathcal{B}_\ddagger$ *and* $\alpha, \beta \in$ $\mathbf{Types}_\ddagger$.

Since terms are considered modulo α-conversion, the weakening rule is admissible. Moreover we have $\Gamma_{|M} \vdash_\dagger M{:}\alpha$ whenever $\Gamma \vdash_\dagger M{:}\alpha$, where $\Gamma_{|M} = \{x{:}\beta \in \Gamma \mid x \in FV(M)\}$ for $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.

We want to consider unions of bases taking the intersections of the types with the same subjects. Since not all intersections of types in $\mathbf{Types}_\mathtt{t}$ 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 20 *Let* $\ddagger \in \{\mathtt{t}, \mathtt{wh}, \mathtt{ei}\}$. *We say that two bases* $\Gamma, \Gamma' \in \mathcal{B}_\ddagger$ *are* compatible *if and only if* $x : \alpha \in \Gamma$ *and* $x : \beta \in \Gamma'$ *imply* $\alpha \wedge \beta \in \mathbf{Types}_\ddagger$. *If* Γ *and* Γ' *are compatible bases we define their union* $\uplus$ *as:*

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

Accordingly we define: $\Gamma \stackrel{+}{\subseteq} \Gamma' \Leftrightarrow \exists \Gamma''. \Gamma \uplus \Gamma'' = \Gamma'$.

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. Due to the presence of rule (Eq_β), in the generation lemma for $\vdash_\mathtt{t}$ we need to consider approximate normal forms instead of arbitrary terms.

Lemma 21 (Generation Lemma for $\vdash_\mathtt{t}$)

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

Lemma 22 (Generation Lemma for $\vdash_\dagger$) *Let* $\dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.

1. $\Gamma \vdash_\dagger \bot{:}\alpha$ *implies* $\alpha =_\dagger \omega$;
2. $\Gamma \vdash_\dagger x{:}\alpha$ *iff* $x : \beta \in \Gamma$ *for some* $\beta \leq_\dagger \alpha$;

3. $\Gamma \vdash_\dagger \lambda x.M{:}\alpha$ *(and* $\alpha \neq_{\mathtt{w}} \omega$ *when* $\dagger = \mathtt{w}$*) iff* $\alpha =_\dagger \bigwedge_{i\in I}(\alpha_i \to \beta_i)$ *and* $\forall i \in I$ $\Gamma, x : \alpha_i \vdash_\dagger M : \beta_i$*;*
4. $\Gamma \vdash_\dagger MN{:}\alpha$ *iff there is* β *such that* $\Gamma \vdash_\dagger M{:}\beta \to \alpha$*, and* $\Gamma \vdash_\dagger N{:}\beta$*.*

With a standard proof we can show that rule (Eq_β) is admissible in the systems $\vdash_\dagger$ for $\dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.

Our type assignment systems enjoy the approximation theorem, i.e. we can deduce a type for a term M iff we can deduce this type for an approximant of M, w.r.t. the relative notion of approximant (Theorem 23). Such a theorem, having an interest also by itself, will be used in the next section to show that our type assignment systems are tools to analyse the observational behaviour represented by trees.

The Approximation Theorem can be proved by means of a variant of Tait's "computability" technique (Tait 1967) by defining sets of "approximable" and "computable" terms.

Theorem 23 (Approximation Theorem) *Let* $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.
$\Gamma \vdash_\dagger M : \alpha$ *iff there is* $A \in \mathcal{A}_\dagger(M)$ *such that* $\Gamma \vdash_\dagger A : \alpha$.

4 CORRESPONDENCE BETWEEN TREES AND TYPINGS

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

We shall provide an (almost) uniform proof for a theorem which considers other trees besides the ones of the results recalled above. More precisely, we shall prove that $\vdash_\dagger$ derives the same types for two terms M, N iff M, N have the same $\dagger$-trees, where $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.
In order to prove this property, we follow an approach similar to (Dezani et al. 1997) and to (Berarducci et al. to appear) 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 us 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 $\mathbf{Types}_{\mathtt{t}}$, $\mathbf{Types}_{\mathtt{wh}}$ and $\mathbf{Types}_{\mathtt{ei}}$. It is easy to verify that each one of these constants belong to the corresponding set of types.

Definition 24 *1. Let* $\theta = (\omega\omega \to \omega \to \omega) \wedge ((\omega \to \omega) \to \omega\omega)$. *We define* ϕ_0 *as the type* $\omega(\omega\omega \to \theta)$ *and inductively* $\phi_{i+1} \equiv \phi_i\theta$.

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* $i \leq n$.

We need to consider special kinds of bases which allow us to distinguish occurrences of different variables or even different occurrences of the same variable. More precisely in 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 25 *1. We define* $\Gamma_{\mathtt{t}} \in \mathcal{B}_{\mathtt{t}}$ *as the basis* $\{x_n{:}\,\phi_n \mid n \in \mathtt{N}\}$.

2. $\Gamma \in \mathcal{B}_{\mathtt{wh}}$ *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 basis and generalized special basis contain only arrow types and atomic types.

We associate now to each approximate normal form $A \in \mathcal{A}_\dagger$ a basis $\Gamma \in \mathcal{B}_\ddagger$ and a type $\gamma \in \mathbf{Types}_\ddagger$ for $\dagger \in \{\mathtt{t}, \mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$. We call the pair $\langle\Gamma;\gamma\rangle$ the $\dagger$-characteristic pair of A.

Definition 26 *Let* $A \in \mathcal{A}_{\mathtt{t}}$.

1. *The* t-characteristic type $ct_{\mathtt{t}}(A)$ *of* A *is defined as follows.*

   (a) $ct_{\mathtt{t}}(\lambda x_i.A) = \phi_i \to ct_{\mathtt{t}}(A)$,
   (b) $ct_{\mathtt{t}}(\bot A_1 \ldots A_n) = \omega ct_{\mathtt{t}}(A_1) \ldots ct_{\mathtt{t}}(A_n)$,
   (c) $ct_{\mathtt{t}}(x_i A_1 \ldots A_n) = \phi_i ct_{\mathtt{t}}(A_1) \ldots ct_{\mathtt{t}}(A_n)$.

2. *The* t-characteristic pair $cp_{\mathtt{t}}(A)$ *of* A *is* $\langle\Gamma_{\mathtt{t}}; ct_{\mathtt{t}}(A)\rangle$.

Definition 27 *Let* $A \in \mathcal{A}_\dagger^{(n)}$ *for* $\dagger \in \{\mathtt{w}, \mathtt{h}, \mathtt{e}, \mathtt{i}\}$.
The $\dagger$-characteristic pair *of degree* n $pp_\dagger^{(n)}(A)$ *of* A *is defined as follows.*

1. *If* $pp_\dagger^{(n)}(A) = \langle\Gamma, x{:}\,\beta;\alpha\rangle$, *then* $pp_\dagger^{(n)}(\lambda x.A) = \langle\Gamma;\beta \to \alpha\rangle$.
2. *If* $pp_\dagger^{(n)}(A) = \langle\Gamma;\alpha\rangle$ *and* x *does not occur in* Γ, *then* $pp_\dagger^{(n)}(\lambda x.A) = \langle\Gamma;\omega \to \alpha\rangle$.
3. *If* $pp_\dagger^{(n)}(A) = \langle\Gamma_i;\alpha_i\rangle$ *where* $i \leq k$ *and* $\Gamma = \biguplus_{i=1}^k \Gamma_i$ *is a special basis of degree* n, *then* $pp_\dagger^{(n)}(\bot A_1 \ldots A_k) = \langle\Gamma;\omega\alpha_1 \ldots \alpha_k\rangle$.

4. *If* $\dagger \in \{\mathtt{w},\mathtt{h}\}$, $pp_\dagger^{(n)}(A_i) = \langle \Gamma_i;\alpha_i\rangle$ *where* $i \leq k$ *and* $\Gamma = \biguplus_{i=1}^{k} \Gamma_i \uplus \{x{:}\alpha_1 \to \ldots \to \alpha_k \to \psi_j^{(n)}\}$ *is a special basis of degree* n*, then* $pp_\dagger^{(n)}(xA_1\ldots A_k) = \langle \Gamma;\psi_j^{(n)}\rangle$. *(In particular when* $k = 0$ *we obtain* $pp_\dagger^{(n)}(x) = \langle\{x{:}\psi_j^{(n)}\};\psi_j^{(n)}\rangle$*).*
5. *If* $\dagger \in \{\mathtt{e},\mathtt{i}\}$, $pp_\dagger^{(n)}(A_i) = \langle \Gamma_i;\alpha_i\rangle$ *where* $i \leq k$ *and* $\Gamma = \biguplus_{i=1}^{k} \Gamma_i \uplus \{x{:}\alpha_1 \to \ldots \to \alpha_k \to \zeta^n \to \chi_j^{(n)}\}$ *is a generalized special basis of degree* n*, then* $pp_\dagger^{(n)}(xA_1\ldots A_k) = \langle \Gamma;\zeta^n \to \chi_j^{(n)}\rangle$. *(In particular when* $k = 0$ *we obtain* $pp_\dagger^{(n)}(x) = \langle\{x{:}\zeta^n \to \chi_j^{(n)}\};\zeta^n \to \chi_j^{(n)}\rangle$*).*

Notice that we can always choose the Γ_i in such a way the conditions of the previous definition are satisfied.

We can now prove that, in all cases, if the $\dagger$-characteristic pair of A is $\langle\Gamma;\gamma\rangle$, and if we deduce $\Gamma \vdash_\dagger B{:}\gamma$ (and some conditions on the number of symbols of B or of $(B)_\dagger^{(h)}$, where h is the height of the tree of A, hold), then $A \sqsubseteq_\dagger B$, where $\sqsubseteq_\dagger$ has been defined in Definition 11.

Lemma 28 *1. If* $\Gamma_\mathtt{t} \vdash_\mathtt{t} A{:}\phi_i$ *then* $A \equiv x_i$.
2. *If* $\Gamma_\mathtt{t} \vdash_\mathtt{t} A{:}\pi\alpha$ *and* $\pi\alpha \neq \phi_i$ *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* $\Gamma_\mathtt{t} \vdash_\mathtt{t} B{:}ct_\mathtt{t}(A)$ *then* $A \sqsubseteq_\mathtt{t} B$.

Lemma 29 *Let* $\dagger \in \{\mathtt{w},\mathtt{h}\}$, $A \in \mathcal{A}_\dagger^{(n)}$ *and* $pp_\dagger^{(n)}(A) = \langle\Gamma;\alpha\rangle$*. Then* $B \in \mathcal{A}_\dagger$ *and* $\Gamma \vdash_\dagger B{:}\alpha$ *imply* $A \sqsubseteq_\dagger B$.

Lemma 30 *Assume* $\dagger \in \{\mathtt{e},\mathtt{i}\}$, $A, B \in \mathcal{A}_\dagger$*. Let* h *be the height of* $\mathcal{T}_\dagger(A)$*, and* n *be such that* $A, (B)_\dagger^{(h)} \in \mathcal{A}_\dagger^{(n)}$*. Then* $pp_\dagger^{(n)}(A) = \langle\Gamma;\alpha\rangle$ *and* $\Gamma \vdash_\dagger B{:}\alpha$ *imply* $A \sqsubseteq_\dagger B$.

Theorem 31 (Main Theorem) *For* $\dagger \in \{\mathtt{t},\mathtt{w},\mathtt{h},\mathtt{e},\mathtt{i}\}$ *the following conditions are equivalent:*

1. $\mathcal{T}_\dagger(M) = \mathcal{T}_\dagger(N)$;
2. $\Gamma \vdash_\dagger M{:}\alpha$ *iff* $\Gamma \vdash_\dagger N{:}\alpha$ *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.

(2.)⇒(1.). If $\mathcal{T}_\dagger(M) \neq \mathcal{T}_\dagger(N)$, then by Lemma 13 we can find an approximate normal form A such that $A \in \mathcal{A}_\dagger(M)$ and there is no $B \in \mathcal{A}_\dagger(N)$ such that $A \sqsubseteq_\dagger B$ (or vice versa). If $\dagger = \mathtt{t}$ we have by the Approximation Theorem and Lemma 28(4.) that $\Gamma_\mathtt{t} \vdash_\mathtt{t} M{:}ct_\mathtt{t}(A)$ and $\Gamma_\mathtt{t} \not\vdash_\mathtt{t} N{:}ct_\mathtt{t}(A)$. If $\dagger \in \{\mathtt{w},\mathtt{h}\}$ let n be so big that $A \in \mathcal{A}_\dagger^{(n)}$ and $\langle\Gamma;\alpha\rangle = pp_\dagger^{(n)}(A)$. We have by the Approximation Theorem and Lemma 29 that $\Gamma \vdash_\dagger M{:}\alpha$ and $\Gamma \not\vdash_\dagger N{:}\alpha$. If $\dagger \in \{\mathtt{e},\mathtt{i}\}$, let h be the height

of $\mathcal{T}_\dagger(A)$ and n be so big that $A, (N)_\dagger^{(h)} \in \mathcal{A}_\dagger^{(n)}$. This implies $(B)_\dagger^{(h)} \in \mathcal{A}_\dagger^{(n)}$ for all $B \in \mathcal{A}(N)$. Moreover let $\langle \Gamma; \alpha \rangle = pp_\dagger^{(n)}(A)$. We have by the Approximation Theorem and Lemma 30 that $\Gamma \vdash_\dagger M{:}\alpha$ and $\Gamma \not\vdash_\dagger N{:}\alpha$. □

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

Acknowledgements. This paper has strongly benefited from comments and remarks by the Anonymous Referees. Matteo Sereno has been of great help in solving some LaTeX compilation problems.
The first author expresses his gratitude to Roberto Nobilio and Valeria Bosso for interesting discussions. The second author thanks Masako Takahashi for hospitality and enlightening discussions. The third author, whose work has been supported by a COE grant of the Science and Technology Agency of Japan, thanks Yoshiki Kinoshita for introducing him to the ETL environment, and Vincent van Oostrom for fruitful discussions.

REFERENCES

Abramsky S.. Domain theory in logical form. *Ann. of Pure and Appl. Logics*, 51, 1991, 1-77.

Abramsky S., Ong C.-H.L.. Full abstraction in the lazy λ-calculus. *Info. and Comp.* 105, 1993, 159-267.

Alessi F. Type preorders. CAAP'94, *LNCS* 787, Springer-Verlag, Berlin, 1994, 37-51.

Barendregt H.. The Lambda Calculus Its Syntax and Semantics. Studies in Logic 103, North-Holland, Amsterdam, 1984.

Barendregt H., Coppo M., Dezani-Ciancaglini M.. A filter lambda model and the completeness of type assignment. *J. Symbolic Logic* 48, 1983, 931-940.

Berarducci A.. Infinite Lambda-calculus and Non-sensible Models. *Logic and Algebra*, Lecture Notes in Pure and Applied Mathematics 180, Marcel Dekker Inc., 1996, 339-378.

Berarducci A., Dezani-Ciancaglini M.. Infinite lambda-calculus and types. *Theor. Comp. Sci.*, to appear.

Boudol G.. A lambda calculus for (strict) parallel functions. *Info. and Comp.* 108, 1994, 51-127.

Boudol G., Laneve C.. The discriminating power of multiplicities in the λ-calculus. *Info. and Comp.* 126(1), 83–102, 1996.

Coppo M., Dezani-Ciancaglini M., Honsell F., Longo G.. Extended type structures and filter lambda models. *Logic Colloquium '82*, North-Holland, Amsterdam, 1983, 241-262.

Coppo M., Dezani-Ciancaglini M., Venneri B.. Principal type schemes and λ-calculus semantics. *To H.B.Curry, Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, 1980, 535-560.

Coppo M., Dezani-Ciancaglini M., Zacchi M.. Type theories, normal forms and $\mathcal{D}_\infty$ lambda models. *Info. and Control* 72(2), 1987, 85-116.

Dezani-Ciancaglini M., de'Liguoro U., Piperno A.. Filter models for conjunctive-disjunctive λ-calculus. *Theor. Comp. Sci.* 170(1-2), 1996, 83–128.

Dezani-Ciancaglini M., de'Liguoro U., Piperno A.. A filter model for concurrent λ-calculi. *SIAM J. of Comp.*, to appear.

Dezani-Ciancaglini M., Tiuryn J., Urzyczyn P.. Discrimination by parallel observers. *LICS '97*, IEEE Comp.Soc. Press, Los Alamitos, 1997, 396–407.

Engeler E.. Algebra and combinators. *Algebra Universalis* 13(3), 1981, 389-392.

Hyland M.. A syntactic characterization of the equality in some models of the λ-calculus. *J. London Math.Soc.* 12(2), 1976, 361-370.

Kennaway J. R., Klop J. W Sleep R., de Vries F.J.. Transfinite reductions in orthogonal term rewriting systems. *Info. and Comp.* 119(1), 1995, 18-38.

Kennaway R., Klop J. W., Sleep R., de Vries F.J.. Infinitary lambda calculus. *Theor. Comp. Sci.*, 175(1), 1997, 93-126.

Kennaway J.R., van Oostrom V., de Vries F.J.. Meaningless terms in rewriting. ALP'96, *LNCS* 1139, Springer-Verlag, Berlin, 1996, 254-268.

Lévy J.J.. An algebraic interpretation of the $\lambda - \beta - \mathbf{K}$-calculus and an application of the labelled λ-calculus. *Theor. Comput. Sci.* 2(1), 97–114, 1976.

Longo G.. Set theoretical models of lambda calculus: theory, expansions and isomorphisms. *Ann. of Pure and Appl. Logic* 24, 153-188, 1983.

Nakajima R.. Infinite normal forms for the λ-calculus. LCCST, LNCS 37, Springer-Verlag, Berlin, 1975, 62-82

Ronchi della Rocca S.. Characterization theorems for a filter lambda model. *Info. and Control* 54, 1982, 201-216.

Sangiorgi D.. "The lazy λ-calculus in a concurrency scenario. *Info. and Comp.* 111(1), 120-153, 1994.

Scott D.S.. Continuous lattices. TAGL, *LNM* 274, Springer-Verlag, Berlin, 1972, 97-136.

Scott D.S.. Data types as lattices. *Siam J. Comput.* 5, 1976, 522-587.

Scott D.S.. Domains for denotational semantics. ICALP'82, *LNCS* 140, Springer-Verlag, Berlin, 1982, 577-613.

Tait W.W.. Intensional interpretation of functionals of finite types I. *J. Symbolic Logic* 32, 1967, 198-212.

Wadsworth C.P.. The relation between computational and denotational properties for scott's D_∞ models of the λ-calculus. *Siam J. Comput.* 5, 1976, 488-521.

BIOGRAPHY

FRANCO BARBANERA was born in Latina (3 Nov. 1963). Ph.D. at the University of Torino. Assistant professor at the University of Torino since 1992.

MARIANGIOLA DEZANI-CIANCAGLINI was born in Torino (22 Dec. 1946). Ph.D. at the University of Nijmegen. Full professor at the University of Torino since 1981. Member of IFIP W.G.2.2 on "Formal Description of Programming Concepts", of the Editorial Board of "Information and Computation", of the "Academia Europaea" and of LICS Advisory Board.

FER-JAN DE VRIES received his Ph.D from the University of Utrecht in 1989. After a six year stay at CWI in Amsterdam he went to Japan as a visiting researcher at NTT and Hitachi. Currently he is projectleader of the Rewriting Group of the El.Lab. in Tsukuba, Japan.