Lambda Models Characterizing Computational Behaviours of Terms

Mariangiola Dezani-Ciancaglini[*]

Dipartimento di Informatica, Università di Torino, Torino, Italy

e-mail: `dezani@di.unito.it`

Silvia Ghilezan[†]

Faculty of Engineering, University of Novi Sad, Novi Sad, Yugoslavia

e-mail: `gsilvia@uns.ns.ac.yu`

Abstract

We build a lambda model which characterizes completely (persistently) normalizing, (persistently) head normalizing, and (persistently) weak head normalizing terms. This is proved by using the finitary logical description of the model obtained by defining a suitable intersection type assignment system.

1 Introduction

The aim of this research is to present lambda models which completely characterize computational properties of lambda terms and to prove this by using the finitary logical description of the models obtained by defining suitable intersection type assignment systems.

In this paper we focus on six computational properties of lambda terms and corresponding six sets of lambda terms: the set of *normalizing*, *head normalizing*, *weak head normalizing* lambda terms, and those corresponding to the *persistent* versions of such notions. We build an *inverse lambda model* $\mathcal{D}_\infty$, according to Scott [20], which completely characterizes each of the six sets of terms mentioned. More precisely for each one of the above six sets of terms there is a corresponding element in the model such that a term belongs to the set if and only if its interpretation (in a suitable environment) is bigger than or equal to that element. This is proved by using the finitary logical description of the model $\mathcal{D}_\infty$ obtained by defining an *intersection type assignment system* in the following way. First, we construct a set $\mathcal{T}$ of types which are generated from

atomic types corresponding to the elements of $\mathcal{D}_0$, by the *function type* constructor and the *intersection type* constructor. Then we define a set $\mathcal{F}$ of filters on the set $\mathcal{T}$. According to Coppo et al. [6] and Alessi [2], the set $\mathcal{F}$ of filters and the inverse model $\mathcal{D}_\infty$ are isomorphic as ω-algebraic lattices. This isomorphism falls in the general framework of *Stone dualities* (Johnstone [12]). This framework later received a categorically principled explanation by Abramsky in the broader perspective of "domain theory in logical form" [1]. The interest of the above isomorphism lies in the fact that the interpretations of lambda terms in $\mathcal{D}_\infty$ are isomorphic to the filters of types one can derive in this type assignment system (Alessi [2]). This gives a desired finitary logical description of the model. Therefore an equivalent of the primary complete characterization can be stated: a term belongs to one of the six sets mentioned if and only if it has a certain type (in a suitable basis) in the type assignment system obtained.

Some ideas of this research are presented in Dezani and Ghilezan [7]. Along the lines of this ideas it is worth investigating the behaviour of some other computational properties such as various closabilities, i.e. reductions to different kinds of closed terms.

In Section 2 the model $\mathcal{D}_\infty$ characterizing all six computational properties of terms is built. The type assignment system is defined in Section 3 and the proofs are shortly discussed.

2 The Model

We use standard notations for lambda terms and beta reductions.

Definition 2.1 (The set Λ of lambda terms) *The set Λ of (type-free) lambda terms is defined by the following abstract syntax.*

$$
\begin{array}{rcl}
\Lambda & ::= & \mathtt{var} \mid \Lambda\Lambda \mid \lambda\mathtt{var}.\Lambda \\
\mathtt{var} & ::= & x \mid \mathtt{var}'
\end{array}
$$

We use $x, y, z, \ldots, x_1, \ldots$ for arbitrary term variables and $M, N, P, \ldots, M_1, \ldots$ for arbitrary terms. $\mathrm{FV}(M)$ denotes the set of free variables of a term M. By $M[x := N]$ we denote the term obtained by substituting the term N for all the free occurrences of the variable x in M, taking into account that free variables of N remain free in the term obtained.

The axiom of β-*reduction* is $(\lambda x.M)N \rightarrow_\beta M[x := N]$. A term of the form $(\lambda x.M)N$ is called β-*redex*. The transitive reflexive closure of $\rightarrow_\beta$ is denoted by $\twoheadrightarrow_\beta$. A term is *a normal form* if it does not contain β-redexes.

We consider here the following computational behaviours of lambda terms.

Definition 2.2 (Normalization properties)

   i) M has a normal form, $M \in \mathcal{N}$, *if M reduces to a normal form;*

   ii) M has a head normal form, $M \in \mathcal{HN}$, *if M reduces to a term of the form $\lambda \vec{x}.y\vec{M}$ (where possibly y appears in $\vec{x}$);*

iii) M has a weak head normal form, $M \in \mathcal{WN}$, if M reduces to an abstraction or to a term starting with a free variable.

For each of the above properties, we shall consider also the corresponding *persistent* version (see Definition 2.3). *Persistently normalizing* terms have been introduced in Böhm and Dezani [5].

Definition 2.3 (Persistent normalization properties)

i) *A term M is* persistently normalizing, $M \in \mathcal{PN}$, if $M\vec{N} \in \mathcal{N}$ for all terms $\vec{N}$ in $\mathcal{N}$.

ii) *A term M is* persistently head normalizing, $M \in \mathcal{PHN}$, if $M\vec{N} \in \mathcal{HN}$ for all terms $\vec{N}$.

iii) *A term M is* persistently weak normalizing, $M \in \mathcal{PWN}$, if $M\vec{N} \in \mathcal{WN}$ for all terms $\vec{N}$.

Example 2.4 *Let* $\mathbf{I} \equiv \lambda x.x,\ \Delta \equiv \lambda x.xx,\ \mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)),\ \mathbf{K} \equiv \lambda xy.x.$

i) $\lambda x.x\Delta\Delta \in \mathcal{N}$, but $\notin \mathcal{PWN}$, since $(\lambda x.x\Delta\Delta)\mathbf{I} \twoheadrightarrow_\beta \Delta\Delta \notin \mathcal{WN}$.

ii) $\lambda x.y(\Delta\Delta) \in \mathcal{PHN}$, but $\notin \mathcal{N}$.

iii) $\lambda x.x(\Delta\Delta) \in \mathcal{HN}$, but $\notin \mathcal{N}$ and $\notin \mathcal{PWN}$, since $(\lambda x.x(\Delta\Delta))\Delta \rightarrow_\beta \Delta(\Delta\Delta) \notin \mathcal{WN}$.

iv) $\mathbf{YK} \in \mathcal{PWN}$, but $\notin \mathcal{HN}$.

v) $\lambda x.\Delta\Delta \in \mathcal{WN}$, but $\notin \mathcal{HN}$ and $\notin \mathcal{PWN}$, since $(\lambda x.\Delta\Delta)M \twoheadrightarrow_\beta \Delta\Delta \notin \mathcal{WN}$.
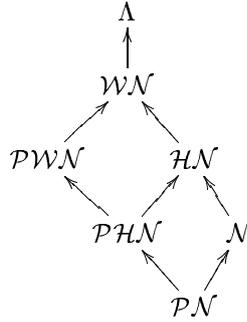

Figure 1: Inclusion between sets of λ-terms

The following proposition, represented pictorially by Figure 1, illustrates mutual implications between the above notions:

Proposition 2.5 *The following* strict *inclusions hold:*

$$
\begin{array}{cccccccc}
\mathcal{PN} & \subsetneq & \mathcal{N} & \subsetneq & \mathcal{HN} & \subsetneq & \mathcal{WN} & \subsetneq & \Lambda \\
\mathcal{PN} & \subsetneq & \mathcal{PHN} & \subsetneq & \mathcal{PWN} & \subsetneq & \mathcal{WN} & & \\
\mathcal{PHN} & \subsetneq & \mathcal{HN}. & & & & &
\end{array}
$$

No other inclusion holds between the above sets.

Our goal is to build an inverse limit lambda model, Scott [20], which satisfies the following condition:
for each one of the above six sets of terms there is a corresponding element in the model such that a term belongs to the set iff its interpretation (in a suitable environment) is bigger than or equal to that element.

We need therefore to discuss the functional behaviours of the terms belonging to these classes, in particular with respect to the step functions, where as usual a step function $\mathsf{a} \Rightarrow \mathsf{b}$ is defined by

$$\lambda \mathsf{d}. \ \textbf{if } \mathsf{a} \sqsubseteq \mathsf{d} \textbf{ then } \mathsf{b} \textbf{ else } \bot.$$

A weak head normalizing term either reduces to an abstraction or to an application of a variable to (possibly zero) terms: in both cases (in a suitable environment) it behaves better than the step function $\bot \Rightarrow \bot$. So we can choose the representative of the step function $\bot \Rightarrow \bot$ as the element which corresponds to $\mathcal{WN}$. We need to consider a model in which this step function is not the bottom of the whole domain, i.e. a solution of the domain equation $D = [D \to D]_\bot$, where as usual $[D \to D]$ is the domain of continuous functions from D to D and $_\bot$ is the lifting operator.

A persistently weak normalizing term applied to any number of arbitrary terms gives a weak head normalizing term: i.e. it behaves better than the step function $\underbrace{\bot \Rightarrow \ldots \Rightarrow \bot}_{n} \Rightarrow \bot$ for all values of n. Therefore the element representing $\bigsqcup_{n \in \mathbb{N}} (\underbrace{\bot \Rightarrow \ldots \Rightarrow \bot}_{n} \Rightarrow \bot)$ is a good candidate for the correspondence with the set $\mathcal{PWN}$.

A head normalizing term when applied to a persistently head normalizing term reduces to a head normalizing term: in its turn a persistently head normalizing term applied to an arbitrary term gives a persistently head normalizing term. Therefore, if h and $\hat{\mathsf{h}}$ are two elements of $\mathcal{D}_0$ corresponding respectively to the sets $\mathcal{HN}$ and $\mathcal{PHN}$, they represent the step functions $\hat{\mathsf{h}} \Rightarrow \mathsf{h}$ and $\bot \Rightarrow \hat{\mathsf{h}}$.

A normalizing term is also a head normalizing term and therefore it behaves better than the step function $\hat{\mathsf{h}} \Rightarrow \mathsf{h}$. A persistently normalizing term is also a persistently head normalizing term and therefore it behaves better than the step function $\bot \Rightarrow \hat{\mathsf{h}}$. A persistently normalizing term applied to a normalizing term gives a persistently normalizing term. Moreover, a normalizing term applied to a persistently normalizing term is in turn a normalizing term.

Therefore if n and $\hat{\mathsf{n}}$ are two elements of $\mathcal{D}_0$ corresponding respectively to the sets $\mathcal{N}$ and $\mathcal{PN}$, they represent the functions $(\hat{\mathsf{h}} \Rightarrow \mathsf{h}) \sqcup (\hat{\mathsf{n}} \Rightarrow \mathsf{n})$ and $(\bot \Rightarrow \hat{\mathsf{h}}) \sqcup (\mathsf{n} \Rightarrow \hat{\mathsf{n}})$.

To sum up we define our model as follows.

4

Definition 2.6 *Let $\mathcal{D}_\infty$ be the inverse limit model obtained by taking as $\mathcal{D}_0$ the lattice of Figure 2, as $\mathcal{D}_1$ the lattice $[\mathcal{D}_0 \to \mathcal{D}_0]_\perp$, and by defining the projection $i_0 : \mathcal{D}_0 \to [\mathcal{D}_0 \to \mathcal{D}_0]_\perp$ as follows:*

$$i_0(\hat{n}) = (\perp \Rightarrow \hat{h}) \sqcup (n \Rightarrow \hat{n}), i_0(n) = (\hat{h} \Rightarrow h) \sqcup (\hat{n} \Rightarrow n),$$

$$i_0(\hat{h}) = \perp \Rightarrow \hat{h}, i_0(h) = \hat{h} \Rightarrow h, i_0(\perp) = \perp.$$
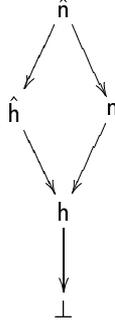

Figure 2: The lattice $\mathcal{D}_0$

Since each variable is clearly a persistently normalizing term, it is meaningful to interpret terms in the environment which maps each variable in the element $\hat{n}$. The main result of our paper is:

Theorem 2.7 (Main Theorem, Version I) *Let $\mathcal{D}_\infty$ be the inverse limit model defined in Definition 2.6 and $\rho_{\hat{n}}$ the environment defined by $\rho_{\hat{n}}(x) = \hat{n}$ for all $x \in \mathtt{var}$. Then:*

i) $M \in \mathcal{PN}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq \hat{n}$;

ii) $M \in \mathcal{N}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq n$;

iii) $M \in \mathcal{PHN}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq \hat{h}$;

iv) $M \in \mathcal{HN}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq h$;

v) $M \in \mathcal{PWN}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq \bigsqcup_{n \in \mathbb{N}}(\underbrace{\perp \Rightarrow \ldots \Rightarrow \perp}_{n} \Rightarrow \perp)$;

vi) $M \in \mathcal{WN}$ *iff* $[\![M]\!]_{\rho_{\hat{n}}}^{\mathcal{D}_\infty} \sqsupseteq \perp \Rightarrow \perp$.

The proof of this theorem is done by means of a finitary logical description of $\mathcal{D}_\infty$ obtained by defining an intersection type assignment system in Section 3.

5

3 The Type Assignment System

Stone dualities allow to describe special classes of topological spaces by means of (possibly finitary) partial orders. Typically, these partial orders are given by the topology, a basis for it, or a subbasis for it. The seminal result is the duality between the categories of Stone spaces and that of Boolean algebras (see Johnstone [12]). Other very important examples are the descriptions of ω-*algebraic complete lattices* as *intersection type theories* in Coppo et al. [6], *Scott domains* as *information systems* in Scott [22], and *SFP domains* as *pre-locales* in Abramsky [1]. It is worthwhile to mention also Martin-Löf's domain interpretation of intuitionistic type theory in Martin-Löf [15].

As stated first in Coppo et al. [6] and proved in Alessi [2], we can describe an inverse limit model $\mathcal{D}_\infty$ by taking:

- the types freely generated by closing (a set of atomic types corresponding to) the elements of $\mathcal{D}_0$ under the *function type* constructor $\rightarrow$ and the *intersection type* constructor $\cap$;

- the preorder between types induced by reversing the order in $\mathcal{D}_0$ and by encoding the initial projection, according to the correspondence:

function type constructor	$\mapsto$	step function
intersection type constructor	$\mapsto$	join.

For the $\mathcal{D}_\infty$ model discussed in previous Section we obtain the following definitions.

Definition 3.1 (The set $\mathcal{T}$ of types) *The set $\mathcal{T}$ of types is defined as follows.*

$$\mathcal{T} \quad ::= \quad \nu \mid \hat{\nu} \mid \mu \mid \hat{\mu} \mid \Omega \mid \mathcal{T} \rightarrow \mathcal{T} \mid \mathcal{T} \cap \mathcal{T}$$

Types will be denoted by $\sigma, \tau, \ldots, \sigma_1, \ldots$. When writing types we shall use the following convention: the constructor $\cap$ takes precedence over the constructor $\rightarrow$ and it associates to the right.

We give now the correspondence between types and elements of $\mathcal{D}_\infty$ (as usual we identify elements of $\mathcal{D}_n$ with their projections in $\mathcal{D}_\infty$).

Definition 3.2 (The mapping m) *The mapping* $\mathtt{m} : \mathcal{T} \rightarrow \mathcal{D}_\infty$ *is defined as follows.*

$$\mathtt{m}(\nu) = \mathtt{n} \quad \mathtt{m}(\hat{\nu}) = \hat{\mathtt{n}} \quad \mathtt{m}(\mu) = \mathtt{h} \quad \mathtt{m}(\hat{\mu}) = \hat{\mathtt{h}} \quad \mathtt{m}(\Omega) = \bot$$
$$\mathtt{m}(\sigma \rightarrow \tau) = \mathtt{m}(\sigma) \Rightarrow \mathtt{m}(\tau) \qquad \mathtt{m}(\sigma \cap \tau) = \mathtt{m}(\sigma) \sqcup \mathtt{m}(\tau).$$

Definition 3.3 (Preorder on $\mathcal{T}$) *The relation $\leq$ is defined on $\mathcal{T}$ by the following axioms and rules:*

$(\hat{\nu}\nu)$	$\hat{\nu} \leq \nu$	$(\hat{\nu}\hat{\mu})$	$\hat{\nu} \leq \hat{\mu}$
$(\nu\mu)$	$\nu \leq \mu$	$(\hat{\mu}\mu)$	$\hat{\mu} \leq \mu$
(Ω)	$\sigma \leq \Omega$	$(\Omega\to)$	$\sigma \to \Omega \leq \Omega \to \Omega$
$(\hat{\nu}\to)$	$\hat{\nu} \sim (\Omega \to \hat{\mu}) \cap (\nu \to \hat{\nu})$	$(\nu\to)$	$\nu \sim (\hat{\mu} \to \mu) \cap (\hat{\nu} \to \nu)$
$(\hat{\mu}\to)$	$\hat{\mu} \sim \Omega \to \hat{\mu}$	$(\mu\to)$	$\mu \sim \hat{\mu} \to \mu$
(refl)	$\sigma \leq \sigma$	*(mon)*	$\sigma \leq \sigma', \ \tau \leq \tau' \Rightarrow \sigma \cap \tau \leq \sigma' \cap \tau'$
(idem)	$\sigma \leq \sigma \cap \sigma$	*(trans)*	$\sigma \leq \tau, \ \tau \leq \rho \Rightarrow \sigma \leq \rho$
(incl)	$\sigma \cap \tau \leq \sigma, \ \sigma \cap \tau \leq \tau$	*(amon)*	$\sigma \leq \sigma', \ \tau \leq \tau' \Rightarrow \sigma' \to \tau \leq \sigma \to \tau'$
		(arint)	$(\sigma \to \tau) \cap (\sigma \to \zeta) \leq \sigma \to \tau \cap \zeta$

where $\sigma \sim \tau$ is short for $\sigma \leq \tau$ and $\tau \leq \sigma$.

The first four axioms correspond to the partial order in $\mathcal{D}_0$. Axiom (Ω) says that Ω is the top type, and therefore gives for Ω both the partial order in $\mathcal{D}_0$ and the projection in $\mathcal{D}_1$. The axioms $(\hat{\nu}\to), (\nu\to), (\hat{\mu}\to), (\mu\to)$ encode the initial projection of the constants different from Ω in $\mathcal{D}_1$. The remaining axioms are standard properties of joins and step functions.

We build filters on the set of types: these will correspond to the elements of $\mathcal{D}_\infty$.

Definition 3.4 (The set $\mathcal{F}$ of filters)　　*i) A* filter *is a set $X \subseteq \mathcal{T}$ such that:*

> *(a) $\Omega \in X$;*
>
> *(b) if $\sigma \leq \tau$ and $\sigma \in X$, then $\tau \in X$;*
>
> *(c) if $\sigma, \tau \in X$, then $\sigma \cap \tau \in X$;*

> *ii) $\mathcal{F}$ denotes the set of filters over $\mathcal{T}$;*
>
> *iii) if $X \subseteq \mathcal{T}$, $\uparrow X$ denotes the filter generated by X;*
>
> *iv) a filter is* principal *if it is of the shape $\uparrow \{\sigma\}$, for some type σ. We shall denote $\uparrow \{\sigma\}$ simply by $\uparrow \sigma$.*

It is easy to verify that the set $\mathcal{F}$ of filters, ordered by subset inclusion, is an ω-algebraic complete lattice, where $\uparrow \Omega$ is the bottom, and $\mathcal{T}$ is the top. Moreover the finite elements are exactly the principal filters.

Using the mapping m we can show that $\mathcal{F}$ and $\mathcal{D}_\infty$ are isomorphic as ω-algebraic complete lattices, as proved (for a general case) in Alessi [2].

Theorem 3.5 (Isomorphism) *The mapping $\mathtt{m}^* : \mathcal{F} \to \mathcal{D}_\infty$ defined by*

$$\mathtt{m}^*(X) = \bigsqcup_{\sigma \in X} \mathtt{m}(\sigma)$$

is a lattice isomorphism between $\mathcal{F}$ and $\mathcal{D}_\infty$.

Notice that $\mathtt{m}^*(\uparrow \sigma) = \mathtt{m}(\sigma)$.

Due to the above isomorphism the interpretations of lambda terms in $\mathcal{D}_\infty$ are isomorphic to the filters of types one can derive in the following type assignment system (Alessi [2]). This gives us a finitary logical description of the model.

A *type assignment* is an expression of the form $M : \tau$, where $M \in \Lambda$ is the *subject* and $\tau \in \mathcal{T}$ is the *predicate*. A *context* Γ is a (possibly infinite) set of type assignments of the shape $x : \sigma$ with different subjects (term variables).

Definition 3.6 (The type assignment system) *The type assignment $M : \tau$ is derivable from the context Γ, notation $\Gamma \vdash M : \tau$, if $\Gamma \vdash M : \tau$ can be generated by the following axioms and rules.*

$$
\frac{}{\Gamma, x : \sigma \vdash x : \sigma} \, (ax) \qquad\qquad \frac{}{\Gamma \vdash M : \Omega} \, (\Omega)
$$

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

$$
\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \cap \tau} \, (\cap I) \qquad \frac{\Gamma \vdash M : \sigma, \ \sigma \leq_\nabla \tau}{\Gamma \vdash M : \tau} \, (\leq)
$$

Theorem 3.7 (Finitary logical description) *For any lambda term M and environment $\rho : \mathtt{var} \to \mathcal{D}_\infty$,*

$$
[\![M]\!]_\rho^{\mathcal{D}_\infty} = \mathtt{m}^*(\{\tau \in \mathcal{T} \mid \exists \Gamma \models \rho.\, \Gamma \vdash M : \tau\}),
$$

where $\Gamma \models \rho$ if and only if $(x : \sigma) \in \Gamma$ implies $\mathtt{m}(\sigma) \sqsubseteq \rho(x)$.

Theorem 3.7 allows us to rephrase the main theorem of previous section, Theorem 2.7, as follows:

Theorem 3.8 (Main Theorem, Version II) *Let $\Gamma_{\hat\nu}$ be the context defined by $\Gamma_{\hat\nu} = \{x : \hat\nu \mid \forall x \in \mathtt{var}\}$. Then:*

   i) $M \in \mathcal{PN}$ iff $\Gamma_{\hat\nu} \vdash M : \hat\nu$;

  ii) $M \in \mathcal{N}$ iff $\Gamma_{\hat\nu} \vdash M : \nu$;

 iii) $M \in \mathcal{PHN}$ iff $\Gamma_{\hat\nu} \vdash M : \hat\mu$;

 iv) $M \in \mathcal{HN}$ iff $\Gamma_{\hat\nu} \vdash M : \mu$;

  v) $M \in \mathcal{PWN}$ iff $\Gamma_{\hat\nu} \vdash M : \Omega^n \to \Omega$ for all $n \in \mathbb{N}$;

 vi) $M \in \mathcal{WN}$ iff $\Gamma_{\hat\nu} \vdash M : \Omega \to \Omega$.

The proofs of the *if* parts of this Theorem are mainly straightforward inductions and case split, and follow, but the case of persistently normalizing terms. For that reason we develop a new characterization of these terms which is less general but much simpler than the one presented in Dezani et al. [8].

The proofs of the *only if* parts require the set-theoretic semantics of intersection types using saturated sets. The *reducibility method* method is a generally accepted way for proving the strong normalization property of various type systems (Tait [23], Tait [24], Girard [11], Mitchell [16] [17],Leivant [14], Gallier [9], Pottinger [19], Krivine [13], van Bakel [25], Ghilezan [10], Amadio and Curien [3], Leivant [14], Gallier [9], Dezani et al. [8]).

In all these papers different properties are characterized by means of different type assignment systems: so the novelty of the present approach is that we characterize all six computational properties of terms by means of a *unique type assignment system*, which induces a λ-model. Moreover in all the papers mentioned different computational properties requires different type interpretations in the reducibility method, whereas we adapt the reducibility method using a *single type interpretation* for all six computational properties.

Lastly we remark that there are essentially *two* semantics for intersection types in the literature and that here we deal with both of them. The *set-theoretical* semantics, originally introduced in by Barendregt et al. [4], generalizes the one given by Scott for simple types (Scott [21]). The meanings of types are subsets of the domain of discourse, arrow types are defined as *logical predicates* and intersection is set-theoretic intersection. This semantics is at the basis of our application of the reducibility method. The second semantics views types as *compact elements* of Plotkin's λ-structures (Plotkin [18]). According to this interpretation, the universal type denotes the least element, intersections denote joins of compact elements, and arrow types allow to internalize the space of continuous endomorphisms. This semantics allows us to obtain the isomorphism between the model $\mathcal{D}_\infty$ and the the set $\mathcal{F}$ of filters of types.

References

[1] Samson Abramsky. Domain theory in logical form. *Ann. Pure Appl. Logic*, 51(1-2):1–77, 1991.

[2] Fabio Alessi. *Strutture di tipi, teoria dei domini e modelli del lambda calcolo*. PhD thesis, Torino University, 1991.

[3] Roberto M. Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*. Cambridge University Press, Cambridge, 1998.

[4] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. Symbolic Logic*, 48(4):931–940 (1984), 1983.

[5] Corrado Böhm and Mariangiola Dezani-Ciancaglini. λ-terms as total or partial functions on normal forms. In λ-*calculus and computer science theory*, vol-

ume 37 of *Lecture Notes in Computer Science*, pages 96–121, Berlin, 1975. Springer.

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

[7] Mariangiola Dezani-Ciancaglini and Silvia Ghilezan. A lambda model characterizing computational behaviours of terms. In Y.Toyama, editor, *International Workshop on Rewriting in Proof and Computation, RPC'01*, pages 100–119, 2001.

[8] Mariangiola Dezani-Ciancaglini, Furio Honsell, and Yoko Motohama. Compositional characterization of λ-terms using intersection types. In *Mathematical Foundations of Computer Science 2000*, volume 1893 of *Lecture Notes in Computer Science*, pages 304–313. Springer, 2000.

[9] Jean Gallier. Typing untyped λ-terms, or reducibility strikes again! *Ann. Pure Appl. Logic*, 91:231–270, 1998.

[10] Silvia Ghilezan. Strong normalization and typability with intersection types. *Notre Dame J. Formal Logic*, 37(1):44–52, 1996.

[11] Jean-Yves Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'elimination des coupures dans l'analyse et la théorie des types. In *2nd Scandinavian Logic Symposium*, pages 63–92. North-Holland, 1971.

[12] Peter T. Johnstone. *Stone spaces*. Cambridge University Press, Cambridge, 1986. Reprint of the 1982 edition.

[13] Jean-Louis Krivine. *Lambda-calcul types et modèles*. Masson, Paris, 1990.

[14] Daniel Leivant. Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68, 1986.

[15] Per Martin-Löf. Lecture notes on domain interpretation of type theory. Programming Methodology Group, Workshop on the Semantics of Programming Languages, Chalmers University of Technology, 1983.

[16] John C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 415–431. Elsevire, Amsterdam, 1990.

[17] John C. Mitchell. *Foundation for Programmimg Languages*. MIT Press, 1996.

[18] Gordon D. Plotkin. Set-theoretical and other elementary models of the λ-calculus. *Theoret. Comput. Sci.*, 121(1-2):351–409, 1993.

[19] Garrel Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 561–577. Academic Press, London, 1980.

[20] Dana S. Scott. Continuous lattices. In *Toposes, algebraic geometry and logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136, Berlin, 1972. Springer.

[21] Dana S. Scott. Open problem. In *Lambda Calculus and Computer Science Theory*, volume 37 of *Lecture Notes in Computer Science*, page 369. Springer, Berlin, 1975.

[22] Dana S. Scott. Domains for denotational semantics. In *Automata, languages and programming*, volume 140 of *Lecture Notes in Mathematics*, pages 577–613, Berlin, 1982. Springer.

[23] William W. Tait. Intensional interpretations of functionals of finite type I. *Journal of Symbolic Logic*, 32:198–212, 1967.

[24] William W. Tait. A realizability interpretation of the theory of species. In *Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, pages 240–251. Springer, 1975.

[25] Steffen van Bakel. Complete restrictions of the intersection type discipline. *Theoret. Comput. Sci.*, 102(1):135–163, 1992.