Electronic Notes in Theoretical Computer Science 84 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume84.html 15 pages

.30

Intersection Types and Computational Rules

Fabio Alessi [1]

Dip. di Matematica e Informatica Via delle Scienze, 206 33100 Udine (Italy)

Franco Barbanera [2]

Dip. di Matematica e Informatica Viale A.Doria, 6 95125 Catania (Italy)

Mariangiola Dezani-Ciancaglini [3]

Dip. di Informatica Corso Svizzera, 185 10149 Torino (Italy)

Abstract

The invariance of the meaning of a λ-term by reduction/expansion w.r.t. the considered computational rules is one of the minimal requirements for a λ-model. Being the intersection type systems a general framework for the study of semantic domains for the Lambda-calculus, the present paper provides a characterisation of "meaning invariance" in terms of characterisation results for intersection type systems enabling typing invariance of terms w.r.t. various notions of reduction/expansion, like β, η and a number of relevant restrictions of theirs.

1 Introduction

A fundamental notion in any computational model is that of *basic* computational step. For the Lambda-calculus, the computational model at the basis of the functional programming paradigm, such a notion is embodied by the β-reduction rule

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

Even if such a notion of reduction can undergo a deeper analysis, as it has been done in various calculi which "decompose" it in more "atomic" steps (like the calculi of explicit substitution, among which [1], [20]), yet, up to now, no other reduction rule

[1] Email: alessi@dimi.uniud.it
[2] Email: barba@dmi.unict.it Partially supported by MURST project NAPOLI.
[3] Email: dezani@di.unito.it Partially supported by EU-FET project DART ST-2001-33477, and MURST projects COMETA and McTati.

has been widely recognised as the core of the computational process for functional programs.

It is no surprise at all that any reasonable notion of computational step does not change the meaning of the object on which it operates, being a computational process intended simply to make such a meaning more *explicit*. It is hence clear why any classical formalisation of a semantics for the Lambda-calculus is required to be *adequate,* that is to provide the same "meaning" for *convertible* terms $M =_\beta N$, i.e. terms that can be obtained one out of the other by means of a number of β-reductions and β-expansions (where the notion of β-expansion is the obvious inverse of that of β-reduction).
"Meaning preservation" by β-reduction, and by β-expansion, are therefore two very basic requirements whose deep investigation cannot be overlooked in any foundational study of the denotational semantics of functional programs.

Such an investigation is indeed the general context of the present paper. However, it is not even possible to start investigating unless one does not manage to identify a finitary and natural framework where most of the many models proposed in the literature for the Lambda-calculus can be "embedded" and analysed.

Type systems for intersection types are definitely a framework with the qualities we are looking for: they form a class of type assignment systems for the untyped λ-calculus which allow to express, in a natural and finitary way, many of the most important *denotational* properties of terms (as a matter of fact also many relevant *operational* properties can be characterised by means of intersection types).
Indeed intersection types are a powerful tool both for the analysis and the synthesis of λ-models (see *e.g.* [6] [9], [14], [19], [18], [23] and the references there): on the one hand, intersection type disciplines provide finitary inductive definitions of interpretation of λ-terms in models. On the other hand, they are suggestive for the shape the domain model has to have in order to exhibit certain properties, see [9], [19], [4], [5], [13], [12].

Intersection types can be also viewed as a restriction of the domain theory in logical form, see [2], to the special case of modelling pure Lambda-calculus by means of ω-algebraic complete lattices. Many properties of these models can be proved using this paradigm, which goes back to Stone duality.

Different finitary characterisations of models for the Lambda-calculus can be obtained by introducing specific constants, typing rules and type preorders in the basic intersection type assignment system. An element of a particular domain, representing the denotational meaning of a term M, comes out to correspond to the set of types that can be inferred for M.
It is then clear that in the framework of intersection type systems, the requirements of "meaning preservation" by β-reduction and "meaning preservation by β-expansion" can be fully formalised in terms of typing invariance, namely, in type theory terminology, by the so called Subject reduction/expansion properties.

The contribution of the present paper to the investigation of the above mentioned "meaning preservation" requirements, is to try and isolate necessary and sufficient conditions to be satisfied by domains, finitary representable using types, in order the requirements to be met. In our particular perspective this will amount to prove a number of characterisation results for the intersection type systems enjoying Subject β-reduction/expansion properties.

Even if of no real computational meaning, the η-rule (the formalisation of the notion of extensionality) plays a crucial role in denotational semantics. We then show also how to characterise the intersection type systems enjoying Subject η-reduction/expansion properties.

It is worth noticing that there have been devised in the literature also many restrictions of the β rule, in order to formalise particular sorts of computations. Interesting examples of such restrictions are Plotkin's β_v-rule [22], the β-**I**-rule [11] and the β-N-rule [18]. In this paper we shall prove that it is possible to characterise Subject reduction/expansion properties also for such restricted notions of computations.

The paper is structured as follows: in Section 2 we recall the definitions of intersection types and intersection type preorders. We shall briefly recall also the main systems proposed in the literature, in particular those related to the use of intersection types for denotational semantics. We shall also introduce conditions on type preorders to be used in our characterisation results. Section 3 discusses intersection type assignment systems and their properties. In Section 4 our characterisations results will be given. For lack of space many proofs are omitted. They can be found in `www.dmi.unict.it/∼barba/PAPERS/ext-wollic03.ps`.

2 Intersection type languages and type preorders

In this section we shall recall the main notions concerning intersection type languages and type preorders.
Intersection types are syntactical objects built by closing a given set $\mathbb{C}$ of *type atoms* (constants) under the *function type* constructor $\rightarrow$ and the *intersection type* constructor $\cap$.

Definition 2.1 [Intersection Type Language] The *intersection type language* over $\mathbb{C}$, denoted by $\mathbb{T} = \mathbb{T}(\mathbb{C})$ is defined by the following abstract syntax:

$$\mathbb{T} = \mathbb{C} \mid \mathbb{T}\rightarrow\mathbb{T} \mid \mathbb{T} \cap \mathbb{T}.$$

NOTATION. Upper case Roman letters i.e. $A, B, \ldots$, will denote arbitrary types. When writing intersection types we shall use the following convention: the constructor $\cap$ takes precedence over the constructor $\rightarrow$ and $\rightarrow$ associates to the right. For example

$$(A\rightarrow B\rightarrow C) \cap A\rightarrow B\rightarrow C \equiv ((A\rightarrow(B\rightarrow C)) \cap A)\rightarrow(B\rightarrow C).$$

In this paper we shall be concerned with several different intersection type languages arising from taking different sets of type atoms, depending on which typing invariance properties we want to capture. Typical choices for the set of type atoms are $\mathbb{C}_\infty$, a countable set of constants, or finite sets like $\{\Omega, \varphi, \omega\}$ or $\{\nu\}$.

Most of the expressive power of intersection type languages comes from the fact that they are endowed with a *preorder relation*, $\leq$, which induces, on the set of types, the structure of a meet semi-lattice with respect to $\cap$. This appears natural when we think of types as sets of denotations and interpret $\cap$ as set-theoretic intersection, and $\leq$ as set inclusion.

Definition 2.2 [Intersection Type Preorder] An *intersection type preorder* $\Sigma = (\mathbb{C}, \leq)$ is a binary relation $\leq$ on $\mathbb{T}(\mathbb{C})$ satisfying the following set of axioms and rules:

$$
\begin{array}{rlrl}
\text{(refl)} & A \leq A & \text{(idem)} & A \leq A \cap A \\[1em]
\text{(incl}_L\text{)} & A \cap B \leq A & \text{(incl}_R\text{)} & A \cap B \leq B \\[1em]
\text{(mon)} & \dfrac{A \leq A' \quad B \leq B'}{A \cap B \leq A' \cap B'} & \text{(trans)} & \dfrac{A \leq B \quad B \leq C}{A \leq C} \\[1em]
(\Omega)\ \text{if}\ \Omega \in \mathbb{C} & A \leq \Omega & (\nu)\ \text{if}\ \nu \in \mathbb{C} & A \to B \leq \nu
\end{array}
$$

Notation. We will write $A \sim B$ for $A \leq B$ and $B \leq A$.

Axiom (Ω) states that the type preorders containing the constant Ω have a maximal element. It is particularly meaningful when used in combination with the Ω-type assignment systems, which essentially treat Ω as the universal type of all λ-terms (see Definition 3.4).

Axiom (ν) states that ν is above any arrow type. This axiom agrees with the ν-type assignment systems, which treat ν as the universal type of all λ-abstractions (see Definition 3.6). Notice that the role of ν may be played by the type $\Omega \to \Omega$, when Ω is in $\mathbb{C}$. For this reason it is of no use to have at the same time ν and Ω, hence we impose as pragmatic rule that the two constants do not occur together in any $\mathbb{C}$.

Notice that associativity and commutativity of $\cap$ (as always modulo $\sim$) follow easily from the above axioms and rules. For instance, commutativity is immediate:

$$A \cap B \leq (A \cap B) \cap (A \cap B) \leq B \cap A.$$

Being $\cap$ commutative and associative, we will write $\bigcap_{i \leq n} A_i$ for $A_1 \cap \ldots \cap A_n$. Similarly we shall write $\cap_{i \in I} A_i$, where I denotes always a finite set. Moreover we convene that $\cap_{i \in \emptyset} A_i$ is Ω when $\Omega \in \mathbb{C}$.

All the type preorders considered so far in the literature are defined for languages over finite or countable sets of atoms and they are "generated" by recursive sets $\bigtriangledown$ of axioms and rules of the shape $A \leq B$ (where $\bigtriangledown$ it is said to generate $\leq$ when $A \leq B$ holds if and only if it can be derived from the axioms and rules of $\bigtriangledown$ together with those in Definition 2.2). Such generated preorders have been referred to as *type theories*. We will denote them by $\Sigma^{\bigtriangledown} = (\mathbb{C}^{\bigtriangledown}, \leq_{\bigtriangledown})$.

Note that there are only countably many possible $\bigtriangledown$; hence, there are uncountably many preorders which cannot be represented this way. Note also that the corre-

spondence $\bigtriangledown \mapsto \leq_{\bigtriangledown}$ is not injective.
In this paper, instead, we try to be as general as possible, sticking to our notion of type preorder which indeed extends the notion of type preorders usually considered in the literature, where rules (Ω) and (ν) are not taken into account and are instead postulated inside the recursive sets generating the type theory.

Figure 1 shows a list of special purpose axioms and rules which have been considered in the literature, and which we shall briefly discuss in the following.

$(\Omega\text{-}\eta)$	$\Omega \leq \Omega{\to}\Omega$	$(\to\text{-}\cap)$	$(A{\to}B) \cap (A{\to}C) \leq A{\to}B \cap C$
$(\Omega\text{-}\mathit{lazy})$	$A{\to}B \leq \Omega{\to}\Omega$	$(\to\text{-}\cap^{\sim})$	$(A{\to}B) \cap (A{\to}C) \sim A{\to}B \cap C$
(η)	$\dfrac{A' \leq A \quad B \leq B'}{A{\to}B \leq A'{\to}B'}$	$(\eta^{\sim})$	$\dfrac{A' \sim A \quad B \sim B'}{A{\to}B \sim A'{\to}B'}$
$(\omega\text{-}\mathit{Scott})$	$\Omega{\to}\omega \sim \omega$	$(\omega\text{-}\mathit{Park})$	$\omega{\to}\omega \sim \omega$
$(\omega\varphi)$	$\omega \leq \varphi$	$(\varphi{\to}\omega)$	$\varphi{\to}\omega \sim \omega$
$(\omega{\to}\varphi)$	$\omega{\to}\varphi \sim \varphi$	(I)	$(\varphi{\to}\varphi) \cap (\omega{\to}\omega) \sim \omega$

Fig. 1. Possible Axioms and Rules concerning $\leq$.

The meaning of axioms and rules of Figure 1 can be grasped if we take types to denote subsets of a domain of discourse and we look at $\to$ as the function space constructor in the light of Curry-Scott semantics, see [25]. Thus the type $A{\to}B$ denotes the set of *total* functions which map each element of A into an element of B.

Since Ω represents the maximal element, i.e. the whole universe, then $\Omega{\to}\Omega$ is the set of functions which applied to an arbitrary element return again an arbitrary element. Thus, axiom $(\Omega\text{-}\eta)$ expresses the fact that all the objects in our domain of discourse are total functions, i.e. that Ω is equal to $\Omega{\to}\Omega$ [6]. If now we want to capture only those terms which truly represent functions, as we do for example in the lazy λ-calculus, we cannot assume axiom $(\Omega\text{-}\eta)$. One still may postulate the weaker property $(\Omega\text{-}\mathit{lazy})$ to make all functions total [3]. It simply says that an element which is a function, because it maps A into B, maps also the whole universe into itself.

The intended interpretation of arrow types motivates axiom $(\to\text{-}\cap)$, which implies that if a function maps A into B, and the same function maps also A into C, then, actually, it maps the whole A into the intersection between B and C (i.e. into $B \cap C$), see [6].

Rule (η) is also very natural in view of the set-theoretic interpretation. It implies that the arrow constructor is contra-variant in the first argument and covariant in the second one. It is clear that if a function maps A into B, and we take a subset A' of A and a superset B' of B, then this function will map also A' into B', see [6].

The rules $(\to\text{-}\cap^{\sim})$ and $(\eta^{\sim})$ are similar to the rules $(\to\text{-}\cap)$ and (η). They capture properties of the graph models for the untyped lambda calculus, see [23] and [15].

The remaining axioms express peculiar properties of D_∞-like inverse limit models, see [6], [10], [9], [19], [17], [12].

We can introduce now a list of significant intersection type preorders which have been extensively considered in the literature. The order is logical, rather than historical, and some references define the models, others deal with the corresponding filter models: [7], [8], [18], [19], [14], [3], [6], [24], [21], [9], [23], [15], [13].

These preorders are of the form $\Sigma^\nabla = (\mathbb{C}^\nabla, \leq_\nabla)$, with various different names ∇, picked for mnemonic reasons. In Figure 2 we list their sets of constants $\mathbb{C}^\nabla$ and their sets ∇ of extra axioms and rules taken from Figure 1. Here $\mathbb{C}_\infty$ is an infinite set of fresh atoms (i.e. different from $\Omega, \nu, \varphi, \omega$).

$\mathbb{C}^{\mathcal{CD}}$	$= \mathbb{C}_\infty$	$\mathcal{CD}$	$= \emptyset$
$\mathbb{C}^{\mathcal{CDV}}$	$= \mathbb{C}_\infty$	$\mathcal{CDV}$	$= \{(\rightarrow\text{-}\cap), (\eta)\}$
$\mathbb{C}^{\mathcal{HL}}$	$= \{\varphi, \omega\}$	$\mathcal{HL}$	$= \mathcal{CDV} \cup \{(\omega\varphi), (\varphi{\rightarrow}\omega), (\omega{\rightarrow}\varphi)\}$
$\mathbb{C}^{\mathcal{HR}}$	$= \{\varphi, \omega\}$	$\mathcal{HR}$	$= \mathcal{CDV} \cup \{(\omega\varphi), (\omega{\rightarrow}\varphi), (I)\}$
$\mathbb{C}^{\mathcal{EHR}}$	$= \{\nu\}$	$\mathcal{EHR}$	$= \mathcal{CDV} \cup \{(\nu)\}$
$\mathbb{C}^{\mathcal{AO}}$	$= \{\Omega\}$	$\mathcal{AO}$	$= \mathcal{CDV} \cup \{(\Omega), (\Omega\text{-}lazy)\}$
$\mathbb{C}^{\mathcal{BCD}}$	$= \{\Omega\} \cup \mathbb{C}_\infty$	$\mathcal{BCD}$	$= \mathcal{CDV} \cup \{(\Omega), (\Omega\text{-}\eta)\}$
$\mathbb{C}^{\mathcal{S}c}$	$= \{\Omega, \omega\}$	$\mathcal{S}c$	$= \mathcal{BCD} \cup \{(\omega\text{-}Scott)\}$
$\mathbb{C}^{\mathcal{P}a}$	$= \{\Omega, \omega\}$	$\mathcal{P}a$	$= \mathcal{BCD} \cup \{(\omega\text{-}Park)\}$
$\mathbb{C}^{\mathcal{CDZ}}$	$= \{\Omega, \varphi, \omega\}$	$\mathcal{CDZ}$	$= \mathcal{HL} \cup \mathcal{BCD}$
$\mathbb{C}^{\mathcal{P}l}$	$= \{\Omega, \varphi\}$	$\mathcal{P}l$	$= \{(\Omega), (\eta^\sim)\}$
$\mathbb{C}^{\mathcal{E}n}$	$= \{\Omega\} \cup \mathbb{C}_\infty$	$\mathcal{E}n$	$= \mathcal{P}l \cup \{(\rightarrow\text{-}\cap^\sim), (\Omega\text{-}\eta)\}$
$\mathbb{C}^{\mathcal{DHM}}$	$= \{\Omega, \varphi, \omega\}$	$\mathcal{DHM}$	$= \mathcal{BCD} \cup \{(\omega\varphi), (\omega\text{-}Scott), (\omega{\rightarrow}\varphi)\}$

Fig. 2. Particular Atoms, Axioms and Rules.

We define two conditions on type preorders to be used in our characterisation results for rule β.

Definition 2.3 Let $\Sigma = (\mathbb{C}, \leq)$ be a type preorder.

(i) Σ is *beta* iff for all $I, A_i, B_i, C, D \in \mathbb{T}(\mathbb{C})$:
$\bigcap_{i\in I}(A_i{\rightarrow}B_i) \leq C{\rightarrow}D \;\Leftrightarrow\; \bigcap_{i\in J} B_i \leq D$ where $J = \{i\in I \mid C \leq A_i\}$.

(ii) Σ is *ν-sound* iff $\nu \not\leq A{\rightarrow}B$ for all $A,\ B \in \mathbb{T}(\mathbb{C})$.

A few comments on the previous definition. If we look at $\cap$ as representing join and arrow types as representing step functions, then the condition for a type theory

of being beta, is exactly the relation which holds between sups of step functions [16].
The condition of being ν-sound is used to prevent both ν from being a redundant type and from assigning too many types to a λ-abstraction (assigning ν amounts exactly to discriminating an abstraction and nothing more). Notice that Σ is trivially ν-sound when $\nu \not\in \mathbb{C}$.
When $\Sigma = \Sigma^{\bigtriangledown}$, for some $\bigtriangledown$, it is usually possible to prove the conditions defined above by induction on the derivation of the generated clause of the preorder.

Proposition 2.4

(i) *All the type theories of Figure 2 are beta.*

(ii) *All the type theories of Figure 2 are ν-sound.*

Notation. We write "the type preorder Σ validates $\bigtriangledown$" to mean that all axioms and rules of $\bigtriangledown$ are admissible in Σ.

In order to characterise the invariance of typing under η-expansion, we need to introduce a further condition on type preorders, which essentially says that each atomic type either is greater or equal to a type which can be deduced for all terms which are abstractions or it is between two intersections of arrow types strictly related.

Definition 2.5 [Eta Preorders] A type preorder $(\mathbb{C}, \leq)$ is *eta* iff for all $\psi \in \mathbb{C}$ either $\bigcap_{i\in I}(A_i \to B_i) \leq \psi$ for some $I, A_i, B_i \in \mathbb{T}(\mathbb{C})$ such that $B_i \sim \Omega$ for all $i \in I$ or $\nu \leq \psi$ or there exist non empty families of types $\{A_i, B_i\}_{i\in I}$, $\{D_{i,j}, E_{i,j}\}_{j\in J_i}$ in $\mathbb{T}(\mathbb{C})$ such that

$$\bigcap_{i\in I}(A_i \to B_i) \leq \psi \leq \bigcap_{i\in I}(\bigcap_{j\in J_i}(D_{i,j} \to E_{i,j})) \ \&$$
$$\forall i \in I.\ A_i \leq \bigcap_{j\in J_i} D_{i,j} \ \& \ \bigcap_{j\in J_i} E_{i,j} \leq B_i.$$

It is easy to verify that if $(\mathbb{C}, \leq)$ validates $\mathcal{CDV}$ then the condition of the above definition simplifies to the requirement that all atomic types are either greater than $\Omega \to \Omega$ or greater than ν, or they are equivalent to a suitable intersection of arrow types, namely

$$\forall \psi \in \mathbb{C}. \Omega \to \Omega \leq \psi \text{ or } \nu \leq \psi \text{ or } \exists I, \{A_i, B_i\}_{i\in I}.\ \bigcap_{i\in I}(A_i \to B_i) \sim \psi.$$

The following proposition singles out all type preorders of Figure 2 which are eta.

Proposition 2.6 *If $\bigtriangledown \in \{\mathcal{HL}, \mathcal{EHR}, \mathcal{AO}, \mathcal{S}c, \mathcal{P}a, \mathcal{CDZ}, \mathcal{DHM}\}$, then $\Sigma^{\bigtriangledown}$ is an eta preorder.*

3 Intersection Type Assignments

We are now ready to introduce the crucial notion of *intersection type assignment system*. First we need a few preliminary definitions.

Definition 3.1 (i) A *basis* over $\mathbb{C}$ is a set of statements of the shape $x{:}B$, where $B{\in}\mathbb{T}(\mathbb{C})$, all whose variables are distinct.

(ii) An *intersection-type assignment system* relative to $\Sigma = (\mathbb{C}, \leq)$, denoted by $\lambda\cap^{\Sigma}$, is a formal system for deriving judgements of the form $\Gamma \vdash^{\Sigma} M : A$, where the *subject* M is an untyped λ-term, the *predicate* A is in $\mathbb{T}(\mathbb{C})$, and Γ is a basis over $\mathbb{C}$.

$$\text{(Ax)}\quad \Gamma \vdash^{\Sigma}_{\mathcal{B}} x{:}A \qquad \text{if } (x{:}A{\in}\Gamma)$$

$$(\to\text{I})\ \frac{\Gamma, x{:}A \vdash^{\Sigma}_{\mathcal{B}} M : B}{\Gamma \vdash^{\Sigma}_{\mathcal{B}} \lambda x.M : A{\to}B} \qquad (\to\text{E})\ \frac{\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A \to B \quad \Gamma \vdash^{\Sigma}_{\mathcal{B}} N : A}{\Gamma \vdash^{\Sigma}_{\mathcal{B}} MN : B}$$

$$(\cap\text{I})\ \frac{\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A \quad \Gamma \vdash^{\Sigma}_{\mathcal{B}} M : B}{\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A \cap B} \qquad (\leq)\ \frac{\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A \quad A \leq B}{\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : B}$$

Fig. 3. The Axioms and Rules of the Basic Type Assignment System.

Notation. When $\Sigma = \Sigma^{\nabla}$ we shall denote $\lambda\cap^{\Sigma}$ and $\vdash^{\Sigma}$ by $\lambda\cap^{\nabla}$ and $\vdash^{\nabla}$, respectively.

Various type assignment systems can be defined, each of them parametrized w.r.t a Σ=($\mathbb{C}, \leq$). The simplest system is given in the following definition.

Definition 3.2 [Basic Type Assignment System]
Given a type preorder Σ, the axioms and rules of the *basic type assignment system*, denoted by $\lambda\cap^{\Sigma}_{\mathcal{B}}$, for deriving judgements $\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A$, are shown in Figure 3.

Example 3.3 Self-application can be easily typed in $\lambda\cap^{\Sigma}_{\mathcal{B}}$, as follows.

$$\dfrac{\dfrac{\dfrac{x{:}(A{\to}B)\cap A \vdash^{\Sigma}_{\mathcal{B}} x{:}(A{\to}B)\cap A}{x{:}(A{\to}B)\cap A \vdash^{\Sigma}_{\mathcal{B}} x{:}A{\to}B}\,(\leq) \quad \dfrac{x{:}(A{\to}B)\cap A \vdash^{\Sigma}_{\mathcal{B}} x{:}(A{\to}B)\cap A}{x{:}(A{\to}B)\cap A \vdash^{\Sigma}_{\mathcal{B}} x{:}A}\,(\leq)}{x{:}(A{\to}B)\cap A \vdash^{\Sigma}_{\mathcal{B}} xx{:}B}\,(\to\text{E})}{\vdash^{\Sigma}_{\mathcal{B}} \lambda x.xx : (A{\to}B)\cap A{\to}B}\,(\to\text{I})$$

If $\Omega{\in}\mathbb{C}$, a natural choice is to set Ω as the universal type of all λ-terms. This amounts to modify the basic type assignment system by adding a suitable axiom for Ω.

Definition 3.4 [Ω-type Assignment System]
Given a type preorder $\Sigma = (\mathbb{C}, \leq)$ with $\Omega{\in}\mathbb{C}$, the axioms and rules of the Ω-*type assignment system* (denoted $\lambda\cap^{\Sigma}_{\Omega}$), for deriving judgements of the form $\Gamma \vdash^{\Sigma}_{\Omega} M : A$, are those of the basic one, plus the further axiom

$$\text{(Ax-}\Omega\text{)}\quad \Gamma \vdash^{\Sigma}_{\Omega} M : \Omega.$$

Example 3.5 Also non-strongly normalising terms can be typed in $\lambda\cap^{\Sigma}_{\Omega}$ even with a type different from Ω. Note the usage of the axiom $(\text{Ax-}\Omega)$. Let $\Delta \equiv \lambda x.xx$.

$$\dfrac{\dfrac{\dfrac{x{:}A, y{:}\Omega \vdash^{\Sigma}_{\Omega} x : A}{y{:}\Omega \vdash^{\Sigma}_{\Omega} \lambda x.x : A\to A}\,(\to\mathrm{I})}{\vdash^{\Sigma}_{\Omega} \lambda yx.x : \Omega\to A\to A}\,(\to\mathrm{I}) \qquad \vdash^{\Sigma}_{\Omega} \Delta\Delta : \Omega}{\vdash^{\Sigma}_{\Omega} (\lambda yx.x)(\Delta\Delta) : A\to A}\,(\to\mathrm{E})$$

Analogously to the case of Ω, when $\nu\in\mathbb{C}$, it is natural to consider ν as the universal type for abstractions, hence modifying the basic system by the addition of a special axiom for ν.

Definition 3.6 [ν-type Assignment System]
Given a type preorder $\Sigma = (\mathbb{C}, \leq)$ with $\nu\in\mathbb{C}$, the axioms and rules of the *ν-type assignment system* (denoted $\lambda\cap^{\Sigma}_{\nu}$), for deriving judgements of the form $\Gamma \vdash^{\Sigma}_{\nu} M : A$, are those of the basic one, plus the further axiom

$$(\text{Ax-}\nu)\ \ \Gamma \vdash^{\Sigma}_{\nu} \lambda x.M : \nu.$$

Example 3.7 Using axiom $(\text{Ax-}\nu)$ we can again type non-strongly normalising terms, but not the term of Example 3.5, as proved in [14].

$$\dfrac{\dfrac{\dfrac{x{:}A, y{:}\nu \vdash^{\Sigma}_{\nu} x : A}{y{:}\nu \vdash^{\Sigma}_{\nu} \lambda x.x : A\to A}\,(\to\mathrm{I})}{\vdash^{\Sigma}_{\nu} \lambda yx.x : \nu\to A\to A}\,(\to\mathrm{I}) \qquad \vdash^{\Sigma}_{\nu} \lambda z.\Delta\Delta : \nu}{\vdash^{\Sigma}_{\nu} (\lambda yx.x)(\lambda z.\Delta\Delta) : A\to A}\,(\to\mathrm{E})$$

For simplicity we assume the symbols Ω and ν to be reserved for the universal type constants respectively used in the systems $\lambda\cap^{\Sigma}_{\Omega}$ and $\lambda\cap^{\Sigma}_{\nu}$. i.e. we forbid $\Omega\in\mathbb{C}$ or $\nu\in\mathbb{C}$ when we deal with $\lambda\cap^{\Sigma}_{\mathcal{B}}$.

Notation. In the following $\lambda\cap^{\Sigma}$ will range over $\lambda\cap^{\Sigma}_{\mathcal{B}}$, $\lambda\cap^{\Sigma}_{\Omega}$ and $\lambda\cap^{\Sigma}_{\nu}$. More precisely we assume that $\lambda\cap^{\Sigma}$ stands for $\lambda\cap^{\Sigma}_{\Omega}$ whenever $\Omega\in\mathbb{C}$, for $\lambda\cap^{\Sigma}_{\nu}$ whenever $\nu\in\mathbb{C}$, and for $\lambda\cap^{\Sigma}_{\mathcal{B}}$ otherwise. Similarly for $\vdash^{\Sigma}$. If there is no danger of confusion, we often write simply $\mathbb{T}$ for $\mathbb{T}^{\Sigma}$ and $\vdash$ for $\vdash^{\Sigma}$.

The subterm property does not hold in general for $\lambda\cap^{\Sigma}_{\nu}$. In fact $\lambda x.M$ is typable also when M is not typable. Moreover, in $\lambda\cap^{\Sigma}_{\Omega}$ and $\lambda\cap^{\Sigma}_{\nu}$, a judgement $\Gamma \vdash^{\Sigma} M : A$ does not imply $FV(M) \subseteq \Gamma$.

One of the most interesting features of intersection type systems is that of enabling precise characterisation results of many important sets of Lambda-terms, among which the one of Strongly-Normalizing terms. We state in the following theorem such a result that shall be needed in the next section, for a proof see [13].

Theorem 3.8 *(Characterization of strongly normalising terms) A λ-term $M\in\mathsf{SN}$ if and only if for all type preorders Σ there exist $A\in\mathbb{T}$ and a $\mathbb{C}$-basis Γ such that $\Gamma \vdash^{\Sigma}_{\mathcal{B}} M : A$.*

3.1 Admissible Rules

In this subsection we introduce a few relevant properties of intersection types, which we shall need for our characterisation results in the next section.

Many interesting type assignment rules can be proved to be admissible.

Proposition 3.9 (Admissible rules) *For arbitrary intersection type theories Σ the following rules are admissible in the intersection type assignment system $\lambda\cap^{\Sigma}$.*

$$(\cap \mathrm{E}_l)\ \frac{\Gamma \vdash^{\Sigma} M : A \cap B}{\Gamma \vdash^{\Sigma} M : A} \qquad (\cap \mathrm{E}_r)\ \frac{\Gamma \vdash^{\Sigma} M : A \cap B}{\Gamma \vdash^{\Sigma} M : B}$$

$$(\mathrm{W})\ \frac{\Gamma \vdash^{\Sigma} M : A \quad x \notin \Gamma}{\Gamma, x{:}B \vdash^{\Sigma} M : A} \qquad (\mathrm{C})\ \frac{\Gamma, x{:}B \vdash^{\Sigma} M : A \quad \Gamma \vdash^{\Sigma} N : B}{\Gamma \vdash^{\Sigma} M[x := N] : A}$$

$$(\mathrm{S})\ \frac{\Gamma, x{:}B \vdash^{\Sigma} M : A \quad x \notin FV(M)}{\Gamma \vdash^{\Sigma} M : A} \qquad (\leq \mathrm{L})\ \frac{\Gamma, x{:}B \vdash^{\Sigma} M : A \quad C \leq B}{\Gamma, x{:}C \vdash^{\Sigma} M : A} \quad \square$$

In the following we shall freely use the rules of the above Proposition.

3.2 Generation Lemmas

We introduce now a few properties enabling to “reverse” some of the rules of the type assignment systems $\lambda\cap^{\Sigma}$, so as to achieve some form of generation (or inversion) lemmas (see Theorems 3.10 and 3.11).

Such properties are not trivial. For instance, for the arrow elimination rule, in general we can only say that when $\Gamma \vdash^{\Sigma} MN : A$, then there are a non-empty, finite set I and types B_i, C_i, such that for each $i{\in}I$, $\Gamma \vdash^{\Sigma} M : B_i{\to}C_i$, $\Gamma \vdash^{\Sigma} N : B_i$, and moreover $\bigcap_{i\in I} C_i \leq A$. Reasoning similarly on the rule $(\to\mathrm{I})$, one can conclude again that it cannot be reversed. More formally, we get the following theorem.

NOTATION. When we write “...assume $A \not\sim_{\Sigma} \Omega$...” we mean that this condition is always true when we deal with $\vdash^{\Sigma}_{\mathcal{B}}$ and $\vdash^{\Sigma}_{\nu}$, while it must be checked for $\vdash^{\Sigma}_{\Omega}$. Similarly, the condition $\nu \not\leq_{\Sigma} A$ must be checked just for $\vdash^{\Sigma}_{\nu}$.

Theorem 3.10 (Generation Lemma I) *Let Σ be a type preorder.*

(i) *Assume $A \not\sim_{\Sigma} \Omega$. Then $\Gamma \vdash^{\Sigma} MN : A$ iff $\Gamma \vdash^{\Sigma} M : B_i{\to}C_i$, $\Gamma \vdash^{\Sigma} N : B_i$, and $\bigcap_{i\in I} C_i \leq A$ for some I non-empty and $B_i, C_i{\in}\mathbb{T}$.*

(ii) *Assume $\nu \not\leq A$. Then $\Gamma \vdash^{\Sigma} \lambda x.M : A$ iff $\Gamma, x{:}B_i \vdash^{\Sigma} M : C_i$, and $\bigcap_{i\in I}(B_i{\to}C_i) \leq A$ for some I non-empty and $B_i, C_i{\in}\mathbb{T}$.*

Using the properties introduced in Definition 2.3, we can give now a rather powerful generation lemma for $\lambda\cap^{\Sigma}$, which is one of the essential ingredients for the proofs of our results.

Special cases of this theorem have been previously proved in [6], [10], [9], [19], [14].

Notation. We write “the type preorder Σ validates $\bigtriangledown$” to mean that all axioms and rules of $\bigtriangledown$ are admissible in Σ.

Theorem 3.11 (Generation Lemma II) *Let Σ be a type preorder.*

(i) *Assume $A \not\sim \Omega$. Then $\Gamma \vdash^{\Sigma} x : A$ iff $(x{:}B) \in \Gamma$ and $B \leq A$ for some $B \in \mathbb{T}^{\Sigma}$.*

(ii) *Assume $A \not\sim \Omega$ and let Σ validate $\mathcal{CDV}$. Then $\Gamma \vdash^{\Sigma} MN : A$ iff $\Gamma \vdash^{\Sigma} M : B \to A$, and $\Gamma \vdash^{\Sigma} N : B$ for some $B \in \mathbb{T}^{\Sigma}$.*

(iii) *Let Σ be ν-sound and beta. Then $\Gamma \vdash^{\Sigma} \lambda x.M : B \to C$ iff $\Gamma, x{:}B \vdash^{\Sigma} M : C$.*

4 Characterization of Subject Reduction and Expansion

In the literature, to which we have provided many references in the previous sections, many models for the Lambda-calculus and a number of its restrictions have been shown to be finitary representable by means of (intersection) types. We now address the general issue of “meaning preservation” by reduction/expansion, dealt with in the Introduction, by characterising the intersection type systems in which types are preserved under various notions of reductions and expansions: β, η, together with some of their *restrictions* given in the literature, like β_v, β-**I** and β-N.

Let us first recall the definitions of these redexes.

Definition 4.1 [Restricted Redexes]

(i) A redex $(\lambda x.M)N$ is a β_v*-redex* if N is a variable or an abstraction [22].

(ii) A redex $(\lambda x.M)N$ is a β-**I**-*redex* if $x \in FV(M)$ [11].

(iii) A redex $(\lambda x.M)N$ is a β-N-*redex* if $x \notin FV(M)$ and N is a closed strongly normalising term [18].

We introduce rules of the form

$$(\text{R-}exp)\quad \frac{M \to_R N \qquad \Gamma \vdash N : A}{\Gamma \vdash M : A} \qquad\qquad (\text{R-}red)\quad \frac{M \to_R N \qquad \Gamma \vdash M : A}{\Gamma \vdash N : A}$$

where $\to_R$ denotes the reduction relation obtained by restricting the contraction to the set of R-redexes. Admissibility of the above rules in a type assignment is usually referred to as *subject expansion* and *subject reduction*, respectively.

Theorem 4.2 (Characterization of subject β-reduction/expansion)

(i) *If $\Gamma \vdash^{\Sigma} M[x := N] : A$ then $\Gamma \vdash^{\Sigma} (\lambda x.M)N : A$ iff N is typable in the context Γ.*

(ii) *(β-expansion) Rule (β-exp) is admissible in $\lambda\cap^{\Sigma}$ iff the condition of* (i) *holds for all pairs of β-redexes and corresponding β-contracta.*

(iii) *(β-reduction) Rule (β-red) is admissible in $\lambda\cap^{\Sigma}$ iff rule ($\to$I) can be reversed, i.e. for all Γ, M, A, B: $\Gamma \vdash^{\Sigma} \lambda x.M : B \to A \;\Rightarrow\; \Gamma, x{:}B \vdash^{\Sigma} M : A$.*

Proof. (i) ($\Rightarrow$) Clearly if N is not typable in the context Γ then also $(\lambda x.M)N$ has no type in Γ by Theorem 3.10(i).

($\Leftarrow$) Let D be a deduction of $\Gamma \vdash^{\Sigma} M[x := N] : A$ and $\Gamma_i \vdash^{\Sigma} N : B_i$ for $i{\in}I$ be all the statements in D whose subject is N. Without loss of generality we can assume that x does not occur in Γ.
If I is non-empty, notice that $\Gamma \subseteq \Gamma_i$ but $\Gamma \upharpoonright FV(N) = \Gamma_i \upharpoonright FV(N)$ (by $\Gamma \upharpoonright \mathcal{X}$ we denote $\{x : A{\in}\Gamma \mid x{\in}\mathcal{X}\}$). So using rules (S) and ($\cap$I), we have that $\Gamma \vdash^{\Sigma} N : \bigcap_{i\in I} B_i$. Moreover, one can easily see, by induction on M, that $\Gamma, x : \bigcap_{i\in I} B_i \vdash^{\Sigma} M : A$. Thus, by rule ($\to$I), we have $\Gamma \vdash^{\Sigma} \lambda x.M : \bigcap_{i\in I} B_i {\to} A$. Hence, by ($\to$E) we can conclude $\Gamma \vdash^{\Sigma} (\lambda x.M)N : A$.
If I is empty, we get from D a derivation of $\Gamma \vdash^{\Sigma} M : A$ by replacing each N by x. By assumption there exists a B such that $\Gamma \vdash^{\Sigma} N : B$. By rule (W) we get $\Gamma, x : B \vdash^{\Sigma} M : A$ and we can conclude as in previous case.

(ii) The proof by a double induction on $\to_{\beta}$ and on derivations is straightforward.

(iii) ($\Rightarrow$) Assume $\Gamma \vdash^{\Sigma} \lambda x.M : B{\to}A$, which implies $\Gamma, y{:}B \vdash^{\Sigma} (\lambda x.M)y : A$ by rule ($\to$E) for a fresh y. The admissibility of rule (β-red) gives us $\Gamma, y{:}B \vdash^{\Sigma} M[x := y] : A$. Hence $\Gamma, x{:}B \vdash^{\Sigma} M : A$.

($\Leftarrow$) It suffices to show that $\Gamma \vdash^{\Sigma} (\lambda x.M)N : A$ implies $\Gamma \vdash^{\Sigma} M[x := N] : A$. The case $A{\sim}\Omega$ is trivial for $\lambda\cap^{\Sigma}_{\Omega}$. Otherwise by Theorem 3.10(ii), there exists a finite set I and types B_i, C_i such that $\Gamma \vdash^{\Sigma} \lambda x.M : B_i {\to} C_i$, $\Gamma \vdash^{\Sigma} N : B_i$ and $\bigcap_{i\in I} C_i \leq A$. By hypothesis we get $\Gamma, x{:}B_i \vdash^{\Sigma} M : C_i$. Then $\Gamma \vdash^{\Sigma} M[x := N] : C_i$ follows by an application of rule (C), and so we can conclude $\Gamma \vdash^{\Sigma} M[x := N] : A$ using rules ($\cap$I) and ($\leq$). $\square$

Corollary 4.3 *If Σ is ν-sound and beta then rule (β-red) is admissible in $\lambda\cap^{\Sigma}$.*

The rather contrived statement given in Theorem 4.2(i) above is immediately met in $\lambda\cap^{\Sigma}_{\Omega}$, in $\lambda\cap^{\Sigma}_{\mathcal{B}}$ when $x{\in}FV(M)$ and in $\lambda\cap^{\Sigma}_{\nu}$ when N is an abstraction. For restricted β-expansions we can give the following simple conditions on type preorders.

Corollary 4.4

(i) *Rule (β-**I**-exp) is admissible in all $\lambda\cap^{\Sigma}_{\mathcal{B}}$ and $\lambda\cap^{\Sigma}_{\Omega}$, but in no $\lambda\cap^{\Sigma}_{\nu}$.*

(ii) *Rule (β-N-exp) is admissible in all $\lambda\cap^{\Sigma}$.*

(iii) *Rule (β_v-exp) is admissible in all $\lambda\cap^{\Sigma}_{\Omega}$ and $\lambda\cap^{\Sigma}_{\nu}$, provided that in this last case each basis Γ contains a statement for each term variable*[4]*. It never holds in $\lambda\cap^{\Sigma}_{\mathcal{B}}$.*

(iv) *Rule (β-exp) is admissible in all $\lambda\cap^{\Sigma}_{\Omega}$, but never in $\lambda\cap^{\Sigma}_{\mathcal{B}}$ and $\lambda\cap^{\Sigma}_{\nu}$.*

Proof. Each of the four items but (ii) follow from Theorem 4.2. An example showing that (β-**I**-exp) is not admissible in $\lambda\cap^{\Sigma}_{\nu}$ is $\vdash^{\nabla}_{\nu} \lambda x.z : \nu$ and $\not\vdash^{\nabla}_{\nu} (\lambda yx.y)z : \nu$.
Item (ii) is a consequence of Theorem 3.8, stating that each strongly normalising term is typable in all intersection type systems from a suitable basis. So all closed strongly normalising terms are typable in all intersection type systems starting from the empty basis.

For the non admissibility of (β_v-exp) in $\lambda\cap^{\Sigma}_{\mathcal{B}}$ and of (β-exp) in $\lambda\cap^{\Sigma}_{\mathcal{B}}$ and $\lambda\cap^{\Sigma}_{\nu}$, notice that we can always derive $\vdash^{\Sigma} \lambda x.x : A{\to}A$, but by the Generation Lemmas I and II

[4] This assumption is sensible for the call-by-value λ-calculus.

(Theorems 3.10(i) and 3.11(i)) we cannot derive the same type for $(\lambda yx.x)z$ from the empty basis without using (Ax-Ω). $\square$

Notice that there are β-redexes that, without being β-**I**-redexes or β-N-redexes, are typable whenever their contracta are. As an example take $(\lambda x.y)y$.

Theorem 4.5 (Characterization of subject η-reduction/expansion)

(i) *Rule (η-exp) is admissible in $\lambda\cap^{\Sigma}$ iff Σ is eta.*

(ii) *Rule (η-red) is admissible in $\lambda\cap^{\Sigma}_{\mathcal{B}}$ iff Σ validates $\mathcal{CDV}$, in $\lambda\cap^{\Sigma}_{\Omega}$ iff Σ validates $\mathcal{BCD}$, and it is never admissible in $\lambda\cap^{\Sigma}_{\nu}$.*

Proof. (i) ($\Rightarrow$) Let $\diamondsuit\in\mathbb{C}$ be a constant that does not satisfy the first two conditions in Definition 2.5. We can derive $x{:}\diamondsuit \vdash^{\Sigma} x : \diamondsuit$. To derive $x{:}\diamondsuit \vdash^{\Sigma} \lambda y.xy : \diamondsuit$ by Theorem 3.10(ii) we need I, A_i, B_i such that $x{:}\diamondsuit, y{:}A_i \vdash^{\Sigma} xy : B_i$ for all $i\in I$ and $\bigcap_{i\in I}(A_i\to B_i) \leq \diamondsuit$. This implies $B_i\not\sim\Omega$ for all $i\in I$, otherwise the first condition of Definition 2.5 would be satisfied. Now by Theorem 3.10(i) we get $x{:}\diamondsuit, y{:}A_i \vdash^{\Sigma} x : D_{i,j}\to E_{i,j}$, $x{:}\diamondsuit, y{:}A_i \vdash^{\Sigma} y : D_{i,j}$, and $\cap_{j\in J_i}E_{i,j} \leq B_i$ for some $J_i, D_{i,j}, E_{i,j}$. By Theorem 3.11(i) we have $\diamondsuit \leq D_{i,j}\to E_{i,j}$ and $A_i \leq D_{i,j}$ for all $i\in I$ and $j\in J_i$. So we conclude

$$\begin{array}{c}\bigcap_{i\in I}(A_i\to B_i)\leq\diamondsuit \leq \bigcap_{i\in I}(\bigcap_{j\in J_i}(D_{i,j}\to E_{i,j}))\\ \forall i\in I.\ A_i \leq \bigcap_{j\in J_i} D_{i,j}\ \&\ \bigcap_{j\in J_i} E_{i,j} \leq B_i.\end{array}$$

($\Leftarrow$) The proof that $\Gamma \vdash^{\Sigma} M : A$ implies $\Gamma \vdash^{\Sigma} \lambda x.Mx : A$, where x is fresh, is by induction on the structure of A. If A is a type constant, then we use the fact that Σ is eta in order to do the derivation discussed in the proof of ($\Rightarrow$). Suppose that $\Gamma \vdash^{\Sigma} M : \psi$ for some $\psi\in\mathbb{C}$ such that:

$$\begin{array}{c}\bigcap_{i\in I}(A_i\to B_i)\leq\psi \leq \bigcap_{i\in I}(\bigcap_{j\in J_i}(D_{i,j}\to E_{i,j}))\\ \forall i\in I.\ A_i \leq \bigcap_{j\in J_i} D_{i,j}\ \&\ \bigcap_{j\in J_i} E_{i,j} \leq B_i.\end{array}$$

By rule ($\leq$) we can derive $\Gamma \vdash^{\Sigma} M : D_{i,j}\to E_{i,j}$ for all $i\in I, j\in J_i$, and so $\Gamma, x{:}D_{i,j} \vdash^{\Sigma} Mx : E_{i,j}$ by rule ($\to$E). From ($\leq$ L), ($\cap$I) and ($\leq$) we get $\Gamma, x{:}A_i \vdash^{\Sigma} Mx : B_i$ and this implies $\Gamma \vdash^{\Sigma} \lambda x.Mx : A_i\to B_i$ using rule ($\to$I). So we can conclude by ($\cap$I) and ($\leq$) that $\Gamma \vdash^{\Sigma} \lambda x.Mx : \psi$. The other cases are easy.

(ii) ($\Rightarrow$) Let us assume that Σ does not validate axiom ($\to$-$\cap$), i.e. that there are types A, B, C such that $(A\to B)\cap(A\to C) \not\leq A\to B\cap C$. We can derive $x{:}(A\to B)\cap(A\to C) \vdash^{\Sigma}_{\mathcal{B}} \lambda y.xy : A\to B \cap C$ using ($\leq$), ($\to$E), ($\cap$I), and ($\to$I), but $x : A\to B \cap C$ cannot be derived from $x{:}(A\to B) \cap (A\to C)$ by Theorem 3.11(i). Now suppose that Σ does not validate rule (η), i.e. that there are types A, B, C, D such that $A \leq B$ and $C \leq D$ but $B\to C \not\leq A\to D$. We can derive $x{:}B\to C \vdash^{\Sigma}_{\mathcal{B}} \lambda y.xy : A\to D$ using ($\leq$), ($\to$E), and ($\to$I), but $x{:}B\to C \not\vdash^{\Sigma}_{\mathcal{B}} x : A\to D$ by Theorem 3.11(i).
If $\Omega\in\mathbb{C}$ we get $x{:}\Omega \vdash^{\nabla}_{\Omega} \lambda y.xy : \Omega\to\Omega$ by axiom (Ax-Ω) and rule ($\to$I). By Theorem 3.11(i) we can derive $x{:}\Omega \vdash^{\nabla}_{\Omega} x : \Omega\to\Omega$ iff $\Omega \leq \Omega\to\Omega$, i.e. iff Σ validates axiom (Ω-η).
If $\nu\in\mathbb{C}$ we get $\vdash^{\nabla}_{\nu} \lambda y.xy : \nu$ by axiom (Ax-ν), but we cannot derive $x : \nu$ from the empty basis by Theorem 3.11(i).

($\Leftarrow$) We prove that under the given conditions on type preorders $\Gamma \vdash^{\Sigma} \lambda x.Mx : A$ and $x \notin FV(M)$ imply $\Gamma \vdash^{\Sigma} M : A$. We give the proof for $\lambda\cap^{\Sigma}_{\Omega}$, that one for $\lambda\cap^{\Sigma}_{\mathcal{B}}$ being similar and simpler. By Theorem 3.10(ii) $\Gamma \vdash^{\nabla}_{\Omega} \lambda x.Mx : A$ implies that there are I, B_i, C_i such that $\Gamma, x{:}B_i \vdash^{\nabla}_{\Omega} Mx : C_i$ and $\bigcap_{i\in I}(B_i\to C_i) \leq A$. If for some i we get $C_i\sim\Omega$, then we can obtain $B_i{\to}C_i{\sim}\Omega$ by axiom (Ω-η) and rule (η). Therefore we can forget those $B_i{\to}C_i$. Otherwise $\Gamma, x{:}B_i \vdash^{\nabla}_{\Omega} Mx : C_i$ implies by Theorem 3.11(ii) and rule (S) that $\Gamma \vdash^{\nabla}_{\Omega} M : D_i{\to}C_i$, and $\Gamma, x{:}B_i \vdash^{\nabla}_{\Omega} x : D_i$, for some D_i. By Theorem 3.11(i) we get $B_i \leq D_i$, so we can derive $\Gamma \vdash^{\nabla}_{\Omega} M : B_i{\to}C_i$ using rule ($\leq$), since $D_i{\to}C_i \leq B_i{\to}C_i$ by rule (η). Rule ($\cap$I) implies $\Gamma \vdash^{\nabla}_{\Omega} M : \bigcap_{i\in I}(B_i{\to}C_i)$. So we can conclude $\Gamma \vdash^{\nabla}_{\Omega} M : A$ using rule ($\leq$). $\square$

5 Conclusions

We have provided characterization results for intersection type systems enabling typing invariance w.r.t. various notions of reduction/expansion, like β, η and a number of their restrictions.
These characterizations possess an interest *per se* in the syntactical theory of intersection types. However, in our intentions, these results aim at being a first step towards the investigation of possible semantic domains modelling exactly one computational reduction/expansion rule, together with the possibility of combining them in order to get models of complete conversions. All this, of course, exploiting the finitary representability of semantic domains offered by intersection types.

Acknowledgements

The second author wishes to express his thanks to Angela Gravina.

References

[1] M. Abadi, L. Cardelli, P.L. Curien, and J.-J. Levy. Explicit substitutions. *J. of Functional Progr.*, 1(4):375–416, 1991.

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

[3] S. Abramsky and C.-H. Luke Ong. Full abstraction in the lazy lambda calculus. *Inform. and Comput.*, 105(2):159–267, 1993.

[4] F. Alessi and M. Dezani-Ciancaglini. Filter models and easy terms. In *ICTCS'01*, volume 2202 of *LNCS*, pages 17–37. Springer-Verlag, 2001.

[5] F. Alessi and S. Lusin. Simple easy terms. In *ITRS'02*, volume 70 of *ENTCS*. Elsevier, 2002.

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

[7] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Formal Logic*, 21(4):685–693, 1980.

[8] M. Coppo, M. Dezani-Ciancaglini, and B. Venneri. Functional characters of solvable terms. *Z. Math. Logik Grundlag. Math.*, 27(1):45–58, 1981.

[9] M. Coppo, M. Dezani-Ciancaglini, and M. Zacchi. Type theories, normal forms, and D_∞-lambda-models. *Inform. and Comput.*, 72(2):85–116, 1987.

[10] M. Coppo, F. Honsell, M. Dezani-Ciancaglini, and G. Longo. Extended type structures and filter lambda models. In *Logic colloquium '82*, pages 241–262. North-Holland, 1984.

[11] H.B. Curry and R. Feys. *Combinatory Logic*, volume I of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1958.

[12] M. Dezani-Ciancaglini and S. Ghilezan. Two behavioural lambda models. In *TYPES'02*, volume 2646 of *LNCS*, pages 127–147. Springer-Verlag, 2003.

[13] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterization of λ-terms using intersection types. In *MFCS'00*, volume 1893 of *LNCS*, pages 304–313. Springer-Verlag, 2000.

[14] L. Egidi, F. Honsell, and S. Ronchi Della Rocca. Operational, denotational and logical descriptions: a case study. *Fund. Inform.*, 16(2):149–169, 1992.

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

[16] G.K. Gierz, K.H. Hoffmann, K. Keimel, J.D. Mislove, and D.S. Scott. *A Compendium of Continuous Lattices*. Springer-Verlag, 1980.

[17] F. Honsell and M. Lenisa. Some results on the full abstraction problem for restricted lambda calculi. In *MFCS'93*, volume 711 of *LNCS*, pages 84–104. Springer-Verlag, 1993.

[18] F. Honsell and M. Lenisa. Semantical analysis of perpetual strategies in λ-calculus. *Theoret. Comput. Sci.*, 212(1-2):183–209, 1999.

[19] F. Honsell and S. Ronchi Della Rocca. An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus. *J. Comput. System Sci.*, 45(1):49–75, 1992.

[20] F. Kamareddine, A. Rios, and J.B. Wells. Calculi of generalized beta-reduction and explicit substitution: Type free and simply typed versions. *J. of Functional and Logic Progr.*, 5:1–44, 1998.

[21] D. Park. The Y-combinator in Scott's λ-calculus models (revised version). Theory of Computation Report 13, Department of Computer Science, University of Warick, 1976.

[22] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoret. Comput. Sci.*, 1(2):125–159, 1975.

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

[24] D.S. Scott. Continuous lattices. In *Toposes, algebraic geometry and logic*, volume 274 of *LNM*, pages 97–136. Springer-Verlag, 1972.

[25] D.S. Scott. Open problem. In *Lambda Calculus and Computer Science Theory*, volume 37 of *LNCS*, page 369. Springer-Verlag, 1975.