Type preorders and recursive terms

Fabio Alessi [1]

Dipartimento di Matematica e Informatica
Università degli Studi di Udine
Via delle Scienze 208, 33100 Udine (Italy)

Mariangiola Dezani-Ciancaglini [2,3]

Dipartimento di Informatica
Università degli Studi di Torino
corso Svizzera 185, 10149 Torino (Italy)

Research Report UDMI/11/2004/RR

Abstract

We show how to use intersection types for building models of a λ-calculus enriched with recursive terms, whose intended meaning is of *minimal* fixed points. As a by-product we prove an interesting consistency result.

1. Introduction

Intersection types were introduced in the late 70's by Coppo and Dezani [7,8,5], to overcome the limitations of Curry's type discipline. They are a very expressive type language which allows to describe and capture various properties of λ-terms. For instance, they have been used in Pottinger [20] to give the first type theoretic characterisation of *strongly normalizable* terms and in Coppo et al. [9] to capture *persistently normalising terms* and *normalising terms*. See Dezani et al. [10] for a more complete account of this line of research.

In this paper we are concerned with the $\lambda\mathsf{rec}$-calculus, obtained from the standard λ-calculus by adding rec-abstraction, according to the syntax:

$$\mathbf{t} ::= x \mid \mathbf{t}\mathbf{t} \mid \lambda x.\mathbf{t} \mid \mathsf{rec}\, x.\mathbf{t}$$

[1] e-mail: alessi@dimi.uniud.it

[2] e-mail: dezani@di.unito.it

[3] Partially supported by EU within the FET - Global Computing initiative, project DART IST-2001-33477, and MURST Project McTati. The funding bodies are not responsible for any use that might be made of the results presented here.

We denote by $\Lambda\mathsf{rec}$ this set of terms.

The reduction rule for rec-abstraction is

$$(\text{red-}\mathsf{rec}) \qquad \mathsf{rec}\, x.\mathbf{t} \to \mathbf{t}[x := \mathsf{rec}\, x.\mathbf{t}]$$

This rule substantiates the intuition of $\mathsf{rec}\, x.\mathbf{t}$ as a fixed-point of $\lambda x.\mathbf{t}$:

$$\mathsf{rec}\, x.\mathbf{t} = (\lambda x.\mathbf{t})(\mathsf{rec}\, x.\mathbf{t}) \to \mathbf{t}[x := \mathsf{rec}\, x.\mathbf{t}]$$

In a sense our rec operator can be seen as an untyped version of the recursion operator in Plotkin PCF [19] and of the fixed point operator in ML language [16].

In this paper we show that intersection types are expressive enough for building models of $\lambda\mathsf{rec}$-calculus in which rec is interpreted as the *least fixed point operator*. Provided that we work with suitable intersection types structures which satisfies the Generation Theorem (we will use the *Easy* ones), we just have to add to the standard *Type Assignment System* the natural rule induced by the interpretation of $\mathsf{rec}\, x.\mathbf{t}$ as minimal fixed point of $\lambda x.\mathbf{t}$ in order to obtain models for the $\lambda\mathsf{rec}$-calculus. This typing rule is:

$$(\mathsf{rec}) \quad \frac{\Gamma, x{:}A \vdash^{\nabla} \mathbf{t} : B \quad \Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : A}{\Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : B}$$

The justification comes from our aim of building models in which the interpretation of terms is the set of types which can be assigned to them. Therefore in agreement with rule (red-rec) we need to derive the same types for $\mathsf{rec}\, x.\mathbf{t}$ and $\mathbf{t}[x := \mathsf{rec}\, x.\mathbf{t}]$. A typing derivation for $\Gamma \vdash^{\nabla} \mathbf{t}[x := \mathsf{rec}\, x.\mathbf{t}] : B$ in general will contain sub-derivations whose conclusions are $\Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : A_i$, for some $i{\in}I$ since $\mathsf{rec}\, x.\mathbf{t}$ is a subterm of $\mathbf{t}[x := \mathsf{rec}\, x.\mathbf{t}]$. We can easily transform such a derivation in a derivation of $\Gamma, x : A \vdash^{\nabla} \mathbf{t} : B$ where $A = \bigcap_{i \in I} A_i$. Moreover by rule $(\cap\mathrm{I})$ (see Definition 2.6) we get $\Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : A$.

A natural question is why we did not simply interpret rec as a fixed point combinator: we will discuss this in the conclusion since it requires some notational conventions introduced in the paper.

Coppo in [6] proved that the set of λ-terms typable in the intersection type assignment systems properly contains the set of λ-terms typable using the ML let rule [16], with respect to the standard mapping associating $\mathsf{let}\ x = N\ \mathsf{in}\ M$ to $(\lambda x.M)N$. This containment does not hold any more if we map the fixed point operator of ML Fix to our recursive operator rec. In fact the ML typing rule for Fix:

$$(\mathsf{Fix}) \quad \frac{\Gamma, x{:}A \vdash \mathbf{t} : A}{\Gamma \vdash \mathsf{Fix} x.\mathbf{t} : A}$$

allows to derive $\vdash \mathsf{Fix} x.x : A$ for all types A, while our rule (rec) allows to obtain for $\mathsf{rec}\, x.x$ just types equivalent to the universal type Ω. This agrees with the requirement that the interpretation of $\mathsf{rec}\, x.x$ is the bottom element.

Notice that Mycroft letrec typing rule [17], [11] is even more permissive than rule (Fix) and then it increases the sets of types derivable for recursive terms.

Comparing the rules (rec) and (Fix), one can observe that rule (rec) does not require that x and $\mathbf{t}$ are typed with the same type, but it has the demanding condition that the type of x must be derivable for the whole recursive term $\mathsf{rec}\, x.\mathbf{t}$. This implies that each type derivation for a recursive term has to start always by assigning the universal type Ω to it. As a by-product of our construction we can prove an interesting consistency result concerning *simple easy* terms. Roughly speaking, a simple easy term is a term to which one can assign an arbitrary intersection type by suitably extending (in a conservative way) the preorder on types. The key property is that in the extended type preorder the term does not receive "too many" types. Notably, simple easiness implies easiness. We show that given any simple easy term $\mathbf{e}$, we can add to the calculus the countable amount of equations

$$(*_{\mathbf{e}}) \quad \forall \mathbf{t} {\in} \Lambda\mathsf{rec}\, .\ \mathbf{e}(\lambda x.\mathbf{t}) = \mathsf{rec}\, x.\mathbf{t}$$

and the resulting theory $\lambda\mathsf{rec} + (*_{\mathbf{e}})$ is still consistent. The proof uses the technique of [2], which allows to build filter models which equate the interpretation of simple easy terms to suitable domain operators.

This consistency result, proved by semantic tools, contrasts the mainstream of consistency proofs in λ-calculi, which is based on the use of syntactic tools (see Kuper [14] and the references there). As to application of semantic tools, we can mention the papers Zylberajch [21], Baeten and Boerboom [4], Alessi et al. [1], Alessi and Lusin [3], Alessi et al. [2].

2. Intersection Type Assignment Systems

Intersection types are syntactic objects built inductively by closing a given set $\mathbb{C}$ of *type atoms* (ground types) plus the universal type Ω under the *function type* constructor $\to$ and the *intersection* type constructor $\cap$.

2.1. Definition. [Intersection type language] Let $\mathbb{C}$ be a countable set of ground types. The *intersection type language* over $\mathbb{C}$, denoted by $\mathbb{T} = \mathbb{T}(\mathbb{C})$, is defined by the following abstract syntax:

$$\mathbb{T} = \Omega \mid \mathbb{C} \mid \mathbb{T} \to \mathbb{T} \mid \mathbb{T} \cap \mathbb{T}.$$

Notation Upper case Roman letters i.e. $A, B, \ldots$, will denote arbitrary types. Greek letters $\alpha, \beta, \ldots$ will denote constants in $\mathbb{C}$. When writing intersection types we shall use the following conventions:

- the constructor $\cap$ takes precedence over the constructor $\to$;
- the constructor $\to$ associates to the right;
- $\bigcap_{i \in I} A_i$ with $I = \{1, \ldots, n\}$ and $n \geq 1$ is short for $((\ldots(A_1 \cap A_2)\ldots) \cap A_n)$;

- $\bigcap_{i\in I} A_i$ with $I=\emptyset$ is Ω.

Much of the expressive power of intersection type disciplines comes from the fact that types can be endowed with a *preorder relation* $\leq$ which satisfies the axioms and rules $\overline{\bigtriangledown}$ of Figure 1, so inducing the structure of a meet semi-lattice with respect to $\cap$, the top element being Ω. We recall here the notion of *easy* intersection type theory as first introduced in Alessi and Lusin [3].

$$
\begin{array}{llll}
(\mathit{refl}) & A \leq A & (\mathit{trans}) & \dfrac{A\leq B \quad B\leq C}{A\leq C} \\[2ex]
(\mathit{mon}) & \dfrac{A\leq A' \quad B\leq B'}{A\cap B\leq A'\cap B'} & (\mathit{idem}) & A\leq A\cap A \\[2ex]
(\mathit{incl}_L) & A\cap B\leq A & (\mathit{incl}_R) & A\cap B\leq B \\[2ex]
(\to\cap) & (A\to B)\cap(A\to C)\leq A\to B\cap C & (\eta) & \dfrac{A'\leq A \quad B\leq B'}{A\to B\leq A'\to B'} \\[2ex]
(\Omega) & A\leq\Omega & (\Omega\text{-}\eta) & \Omega\leq\Omega\to\Omega
\end{array}
$$

Fig. 1. The axioms and rules of $\overline{\bigtriangledown}$

2.2. DEFINITION. [Easy intersection type theories] Let $\mathbb{T}=\mathbb{T}(\mathbb{C})$ be an intersection type language. The *easy intersection type theory* (EITT for short) $\Sigma(\mathbb{C},\bigtriangledown)$ over $\mathbb{T}$ is the set of all judgements $A\leq B$ derivable from $\bigtriangledown$, where $\bigtriangledown$ is a collection of axioms and rules such that (we write $A\sim B$ for $A\leq B$ & $B\leq A$ and $\bigtriangledown^-$ for $\bigtriangledown\setminus\overline{\bigtriangledown}$):

(i) $\bigtriangledown$ contains the set $\overline{\bigtriangledown}$ of axioms and rules shown in Figure 1;

(ii) $\bigtriangledown^-$ contains only axioms of the following two shapes:

$$\alpha\leq\alpha',$$
$$\alpha\sim\bigcap_{h\in H}(\varphi_h\to E_h),$$

where $\alpha,\alpha'\in\mathbb{C}$, $\varphi_h\in\mathbb{C}\cup\{\Omega\}$, and $E_h\in\mathbb{T}$;

(iii) for each $\alpha\in\mathbb{C}$ there is exactly one axiom in $\bigtriangledown^-$ of the shape $\alpha\sim\bigcap_{h\in H}(\varphi_h\to E_h)$;

(iv) let $\bigtriangledown^-$ contain $\alpha\sim\bigcap_{h\in H}(\varphi_h\to E_h)$ and $\alpha'\sim\bigcap_{k\in K}(\varphi'_k\to E'_k)$. Then $\bigtriangledown^-$ contains also $\alpha\leq\alpha'$ if and only if for each $k\in K$, there exists $h_k\in H$ such that $\varphi'_k\leq\varphi_{h_k}$ and $E_{h_k}\leq E'_k$ are both in $\bigtriangledown^-$.

2.3. EXAMPLE. Taking $\mathbb{C}=\{\omega\}$ and $\bigtriangledown^-=\{\omega\leq\omega\to\omega\}$ we obtain an EITT. Honsell and Ronchi [13] show that this EITT induces a filter λ-model (according to the construction given in Section 3) isomorphic to Park λ-model [18].

Notice that:
(a) since $\Omega \sim \Omega \to \Omega \in \Sigma(\mathbb{C}, \triangledown)$ by (Ω) and $(\Omega\text{-}\eta)$, it follows that all atoms in $\mathbb{C}$ are equivalent to suitable (intersections of) arrow types;
(b) $\cap$ (modulo $\sim$) is associative and commutative;
(c) in the last clause of the above definition E'_k and E_{h_k} must be constant types for each $k \in K$ (but we do not denote them with Greek letters since this is a consequence, not an hypothesis).

Notation When we consider an EITT $\Sigma(\mathbb{C}, \triangledown)$, we will write $\mathbb{C}^\triangledown$ for $\mathbb{C}$, $\mathbb{T}^\triangledown$ for $\mathbb{T}(\mathbb{C})$ and $\Sigma^\triangledown$ for $\Sigma(\mathbb{C}, \triangledown)$. Moreover $A \leq_\triangledown B$ will be short for $(A \leq B) \in \Sigma^\triangledown$ and $A \sim_\triangledown B$ for $A \leq_\triangledown B \leq_\triangledown A$. We will consider syntactic equivalence "$\equiv$" of types up to associativity and commutativity of $\cap$.

A nice feature of EITTs is that the order between intersections of arrows agrees with the order between joins of step functions. This property, which implies the representability of all continuous functions in the induced filter λ-models (see Section 3) and which is fully explained in Section 2 of [10], relies on the next theorem. For a proof see [2].

2.4. Theorem. For all I, and $A_i, B_i, C, D \in \mathbb{T}^\triangledown$,

$$\bigcap_{i \in I}(A_i \to B_i) \leq_\triangledown C \to D \text{ if and only if } \bigcap_{i \in J} B_i \leq_\triangledown D \text{ where } J = \{i \in I \mid C \leq_\triangledown A_i\}.$$

Notice that in the statement of Theorem 2.4 the set J may be empty, and in this case we get $\Omega \sim_\triangledown D$.

Before giving the crucial notion of *intersection-type assignment system*, we introduce bases and some related definitions.

2.5. Definition. [Basis] A $\triangledown$-*basis* is a (possibly infinite) set of statements of the shape $x : A$, where $A \in \mathbb{T}^\triangledown$, with all variables distinct.

We will use the following notation:

(i) $x \in \Gamma$ is short for $(x : A) \in \Gamma$ for some A.
(ii) If Γ is a $\triangledown$-basis and $A \in \mathbb{T}^\triangledown$ then $\Gamma, x : A$ is short for $\Gamma \cup \{x : A\}$ when $x \notin \Gamma$.
(iii) Let Γ and Γ' be $\triangledown$-bases. The $\triangledown$-basis $\Gamma \uplus \Gamma'$ is defined as follows:

$$\begin{aligned} \Gamma \uplus \Gamma' = \ & \{x : A \cap B \mid x : A \in \Gamma \text{ and } x : B \in \Gamma'\} \\ & \cup \ \{x : A \mid x : A \in \Gamma \text{ and } x \notin \Gamma'\} \\ & \cup \ \{x : B \mid x : B \in \Gamma' \text{ and } x \notin \Gamma\}. \end{aligned}$$

2.6. Definition. [Type assignment systems] The *intersection type assignment system* relative to the EITT $\Sigma^\triangledown$, notation $\lambda\cap^\triangledown$, is a formal system for deriving judgements of the form $\Gamma \vdash^\triangledown \mathbf{t} : A$, where the *subject* $\mathbf{t}$ is an untyped λ-term, the *predicate* A is in $\mathbb{T}^\triangledown$, and Γ is a $\triangledown$-basis. Its axioms and rules are the following:

$$(\mathrm{Ax})\ \frac{(x:A)\in\Gamma}{\Gamma\vdash^{\triangledown} x:A} \qquad (\mathrm{Ax}\text{-}\Omega)\ \Gamma\vdash^{\triangledown}\mathbf{t}:\Omega$$

$$(\to\mathrm{I})\ \frac{\Gamma,x:A\vdash^{\triangledown}\mathbf{t}:B}{\Gamma\vdash^{\triangledown}\lambda x.\mathbf{t}:A\to B} \qquad (\to\mathrm{E})\ \frac{\Gamma\vdash^{\triangledown}\mathbf{t}:A\to B\quad \Gamma\vdash^{\triangledown}\mathbf{u}:A}{\Gamma\vdash^{\triangledown}\mathbf{tu}:B}$$

$$(\cap\mathrm{I})\ \frac{\Gamma\vdash^{\triangledown}\mathbf{t}:A\quad\Gamma\vdash^{\triangledown}\mathbf{t}:B}{\Gamma\vdash^{\triangledown}\mathbf{t}:A\cap B} \qquad (\leq_{\triangledown})\ \frac{\Gamma\vdash^{\triangledown}\mathbf{t}:A\quad A\leq_{\triangledown}B}{\Gamma\vdash^{\triangledown}\mathbf{t}:B}$$

$$(\mathsf{rec}\,)\quad \frac{\Gamma,x:A\vdash^{\triangledown}\mathbf{t}:B\quad \Gamma\vdash^{\triangledown}\mathsf{rec}\,x.\mathbf{t}:A}{\Gamma\vdash^{\triangledown}\mathsf{rec}\,x.\mathbf{t}:B}$$

2.7. EXAMPLE. The term $\lambda x.\mathsf{rec}\,y.xy$ can be easily typed in all $\lambda\cap^{\triangledown}$, as follows.

$$\dfrac{\dfrac{\dfrac{x:\Omega\to A,y:\Omega\vdash^{\triangledown}x:\Omega\to A\quad x:\Omega\to A,y:\Omega\vdash^{\triangledown}y:\Omega}{x:\Omega\to A,y:\Omega\vdash^{\triangledown}xy:A}(\to\mathrm{E})\quad x:\Omega\to A\vdash^{\triangledown}\mathsf{rec}\,y.xy:\Omega}{x:\Omega\to A\vdash^{\triangledown}\mathsf{rec}\,y.xy:A}(\mu)}{\vdash^{\triangledown}\lambda x.\mathsf{rec}\,y.xy:(\Omega\to A)\to A}(\to\mathrm{I})$$

Notice that due to the presence of axiom (Ax-Ω), one can type terms without assuming types for all their free variables.

As usual we consider λ-terms modulo α-conversion. Notice that the intersection elimination rules

$$(\cap\mathrm{E})\ \frac{\Gamma\vdash^{\triangledown}\mathbf{t}:A\cap B}{\Gamma\vdash^{\triangledown}\mathbf{t}:A}\qquad \frac{\Gamma\vdash^{\triangledown}\mathbf{t}:A\cap B}{\Gamma\vdash^{\triangledown}\mathbf{t}:B}$$

are derivable[4] in any $\lambda\cap^{\triangledown}$.

Moreover, the following rules are admissible:

$$(\leq_{\triangledown}\mathrm{L})\ \frac{\Gamma,x:A\vdash\mathbf{t}:B\quad A'\leq_{\triangledown}A}{\Gamma,x:A'\vdash\mathbf{t}:B}$$

$$(\mathrm{W})\ \frac{\Gamma\vdash\mathbf{t}:B\quad x\notin\Gamma}{\Gamma,x:A\vdash\mathbf{t}:B}\qquad (\mathrm{S})\ \frac{\Gamma,x:A\vdash\mathbf{t}:B\quad x\notin FV(\mathbf{t})}{\Gamma\vdash\mathbf{t}:B}$$

Before giving the Generation Theorem, which essentially will enable us to "reverse" the rules of the type assignment system, we have to take some care about recursive terms, since in deriving types for them with rule (rec), we have premises that contains the terms themselves. So, in doing proofs by induction on derivations, we need to take into account how many times we applied rule (rec).

[4] Recall that a rule is *derivable* in a system if, for each instance of the rule, there is a deduction in the system of its conclusion from its premises. A rule is *admissible* in a system if, for each instance of the rule, if its premises are derivable in the system then so is its conclusion.

2.8. DEFINITION. Given a derivation $\mathcal{D}$ of the judgment $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$, $\nu(\mathcal{D})$ is the number of applications of rule (rec) in $\mathcal{D}$ which have $\mathsf{rec}\, x.\mathbf{t}$ as subject of the conclusion.

Notice that $\nu(\mathcal{D})$ does not take into account possible applications of (rec) to proper subterms of the term in $\mathcal{D}$.

The first four points of the following Generation Theorem have been shown in [2]: the last point which characterises the rec-terms can be easily checked by induction on derivations.

2.9. THEOREM (Generation Theorem).

(i) Assume $A \not\sim_{\triangledown} \Omega$. Then $\Gamma \vdash^{\triangledown} x : A$ if and only if $(x : B) \in \Gamma$ and $B \leq_{\triangledown} A$ for some $B \in \mathbb{T}^{\triangledown}$.

(ii) $\Gamma \vdash^{\triangledown} \mathbf{tu} : A$ if and only if $\Gamma \vdash^{\triangledown} \mathbf{t} : B \to A$, and $\Gamma \vdash^{\triangledown} \mathbf{u} : B$ for some $B \in \mathbb{T}^{\triangledown}$.

(iii) $\Gamma \vdash^{\triangledown} \lambda x.\mathbf{t} : A$ if and only if $\Gamma, x : B_i \vdash^{\triangledown} \mathbf{t} : C_i$ and $\bigcap_{i \in I}(B_i \to C_i) \leq_{\triangledown} A$, for some I and $B_i, C_i \in \mathbb{T}^{\triangledown}$.

(iv) $\Gamma \vdash^{\triangledown} \lambda x.\mathbf{t} : B \to C$ if and only if $\Gamma, x : B \vdash^{\triangledown} \mathbf{t} : C$.

(v) Let $A \not\sim_{\triangledown} \Omega$. $\mathcal{D}$ is a derivation of $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$ if and only if there exist a non-empty set I, and $\mathcal{D}_i$, C_i, B_i, such that:
- for any $i \in I$, $\mathcal{D}_i$ is a subderivation of $\mathcal{D}$ with conclusion $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : B_i$;
- $\forall i \in I. \Gamma, x : B_i \vdash^{\triangledown} \mathbf{t} : C_i$;
- $\bigcap_{i \in I} C_i \leq_{\triangledown} A$;
- $\Sigma_{i \in I} \nu(\mathcal{D}_i) + |I| = \nu(\mathcal{D})$.

Note that in points (i) and (v) of the previous theorem, we have to suppose that $A \not\sim^{\triangledown} \Omega$, otherwise we could derive $\vdash^{\triangledown} x : \Omega$ and $\vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : \Omega$ just using axiom (Ax-Ω).

2.10. THEOREM (Substitution Lemma). $\Gamma \vdash^{\triangledown} \mathbf{t}[z/\mathbf{u}] : A$ if and only if $\exists D \in \mathbb{T}^{\triangledown}.\ \Gamma, z : D \vdash^{\triangledown} \mathbf{t} : A \ \&\ \Gamma \vdash^{\triangledown} \mathbf{u} : D$.

PROOF. $(\Rightarrow)$ is a consequence of the Generation Theorem, reasoning by induction on the structure of $\mathbf{t}$. We just consider the case of $\mathbf{t} \equiv \mathsf{rec}\, x.\mathbf{v}$. We proceed by a further induction on $\nu(\mathcal{D})$, where $\mathcal{D}$ is the derivation of $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v} : A$. If $\nu(\mathcal{D}) = 0$, then $A \sim_{\triangledown} \Omega$ and the thesis is trivial. Otherwise, by the Generation Theorem, point (v), there exist I, $\mathcal{D}_i$, C_i, B_i such that

(i) for any $i \in I$, $\mathcal{D}_i$ is a subderivation in $\mathcal{D}$ of $\Gamma, z : D \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v} : B_i$;

(ii) $\forall i \in I.\ \Gamma, z : D, x : B_i \vdash^{\triangledown} \mathbf{v} : C_i$;

(iii) $\bigcap_{i \in I} C_i \leq_{\triangledown} A$;

(iv) $\Sigma_{i \in I} \nu(\mathcal{D}_i) + |I| = \nu(\mathcal{D})$.

Applying the induction on $\nu(\mathcal{D})$ to (i) and the induction on terms to (ii), we have, for any $i \in I$,
- $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v}[z/\mathbf{u}] : B_i$;

- $\Gamma, x : B_i \vdash^{\triangledown} \mathbf{v}[z/\mathbf{u}] : C_i$.
Applying (rec) to the last two judgements, we have, for any $i \in I$, $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v}[z/\mathbf{u}] : C_i$. By (iii), rules $(\cap \mathrm{I})$ and $(\leq_{\triangledown})$ we get $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v}[z/\mathbf{u}] : A$.

$(\Leftarrow)$ We reason again by induction on the structure of $\mathbf{t}$, with the nested induction on $\nu(\mathcal{D})$ in the case of $\mathbf{t} = \mathsf{rec}\, x.\mathbf{v}$. We just consider the two cases of application and recursive terms. Let $\Gamma \vdash^{\triangledown} \mathbf{t}[z/\mathbf{u}] : A$, with $\mathbf{t} \equiv \mathbf{t}_1 \mathbf{t}_2$. By the Generation Theorem, point (ii), there exists A' such that $\Gamma \vdash^{\triangledown} \mathbf{t}_1[z/\mathbf{u}] : A' \to A$, $\Gamma \vdash^{\triangledown} \mathbf{t}_2[z/\mathbf{u}] : A'$. By induction, there exist types D_1, D_2, such that:
- $\Gamma, z : D_1 \vdash^{\triangledown} \mathbf{t}_1 : A' \to A$; $\Gamma \vdash^{\triangledown} \mathbf{u} : D_1$;
- $\Gamma, z : D_2 \vdash^{\triangledown} \mathbf{t}_2 : A'$; $\Gamma \vdash^{\triangledown} \mathbf{u} : D_2$.
By rule $(\cap \mathrm{I})$ we get $\Gamma \vdash^{\triangledown} \mathbf{u} : D_1 \cap D_2$.
By rule $(\leq_{\triangledown} \mathrm{L})$ we have $\Gamma, z : D_1 \cap D_2 \vdash^{\triangledown} \mathbf{t}_1 : A' \to A$ and $\Gamma, z : D_1 \cap D_2 \vdash^{\triangledown} \mathbf{t}_2 : A'$. By applying $(\to \mathrm{E})$ it follows $\Gamma, z : D_1 \cap D_2 \vdash^{\triangledown} \mathbf{t}_1 \mathbf{t}_2 : A$. The proof for this case is complete by choosing $D \equiv D_1 \cap D_2$.
Let $\mathcal{D}$ be a derivation of $\Gamma \vdash^{\triangledown} \mathbf{t}[z/\mathbf{u}] : A$, with $\mathbf{t} \equiv \mathsf{rec}\, x.\mathbf{v}$. We reason by induction on $\nu(\mathcal{D})$. If $\nu(\mathcal{D}) = 0$, then $A \sim_{\triangledown} \Omega$ and can choose $D \equiv \Omega$. Otherwise, by the Generation Theorem, point (v), there exist I, $\mathcal{D}_i$, C_i, B_i, such that:

(i) for any $i \in I$, $\mathcal{D}_i$ is a subderivation in $\mathcal{D}$ of $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v}[z/\mathbf{u}] : B_i$;

(ii) $\forall i \in I.\ \Gamma, x : B_i \vdash^{\triangledown} \mathbf{v}[z/\mathbf{u}] : C_i$;

(iii) $\bigcap_{i \in I} C_i \leq_{\triangledown} A$;

(iv) $\Sigma_{i \in I} \nu(\mathcal{D}_i) + |I| = \nu(\mathcal{D})$.

Applying induction to (i), for any $i \in I$, there exist D_i such that $\Gamma, z : D_i \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v} : B_i$ and $\Gamma \vdash^{\triangledown} \mathbf{u} : D_i$. Applying induction to (ii), for any $i \in I$, there exist D_i' such that $\Gamma, z : D_i', x : B_i \vdash^{\triangledown} \mathbf{v} : C_i$ and $\Gamma \vdash^{\triangledown} \mathbf{u} : D_i'$. Let $\tilde{D} \equiv \bigcap_{i \in I}(D_i \cap D_i')$. By rule $(\leq_{\triangledown} \mathrm{L})$ it follows:
- $\forall i \in I.\ \Gamma, z : \tilde{D} \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v} : B_i$ and $\Gamma \vdash^{\triangledown} \mathbf{u} : D_i$;
- $\forall i \in I.\ \Gamma, z : \tilde{D}, x : B_i \vdash^{\triangledown} \mathbf{v} : C_i$.
Applying to these last two judgments rule (rec) followed by rules $(\cap \mathrm{I})$ and $(\leq_{\triangledown} \mathrm{L})$, and using (iii), we get $\Gamma, z : \tilde{D} \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{v} : A$. Applying rule $(\cap \mathrm{I})$ to the judgments concerning $\mathbf{u}$, we get $\Gamma \vdash^{\triangledown} \mathbf{u} : \tilde{D}$. Type $\tilde{D}$ allows to prove the thesis.

We end the section with an important consequence of the Substitution Lemma, namely the invariance of type assignment with respect to rule (red-rec): a type is derivable for $\mathsf{rec}\, x.\mathbf{t}$ if and only if it is derivable for $\mathbf{t}[x/\mathsf{rec}\, x.\mathbf{t}]$.

2.11. PROPOSITION. $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$ if and only if $\Gamma \vdash^{\triangledown} \mathbf{t}[x/\mathsf{rec}\, x.\mathbf{t}] : A$.

PROOF. $(\Rightarrow)$ Suppose $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$. Then by the Generation Theorem, point (v), there exists I, B_i, C_i, such that, for any $i \in I$, $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : B_i$, $\Gamma, x :: B_i \vdash^{\triangledown} \mathbf{t} : C_i$ and moreover $\bigcap_{i \in I} C_i \leq_{\triangledown} A$. By the Substitution Lemma we get, for any $i \in I$, $\Gamma \vdash^{\triangledown} \mathbf{t}[x/\mathsf{rec}\, x.\mathbf{t}] : C_i$, hence, using $(\cap \mathrm{I})$ and $(\leq_{\triangledown} \mathrm{L})$, we get $\Gamma \vdash^{\triangledown} \mathbf{t}[x/\mathsf{rec}\, x.\mathbf{t}] : A$.

($\Leftarrow$) Suppose $\Gamma \vdash^{\nabla} \mathbf{t}[x/\mathsf{rec}\, x.\mathbf{t}] : A$. By the Substitution Lemma, there exists D such that $\Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : D$ and $\Gamma, x{:}D \vdash^{\nabla} \mathbf{t} : A$. By applying rule $(\mathsf{rec}\,)$ we obtain $\Gamma \vdash^{\nabla} \mathsf{rec}\, x.\mathbf{t} : A$.

3. Filter Models

Before entering the details of how to build models out of intersection types, we have to focus on the precise notion of model for $\lambda\mathsf{rec}$-calculus. Actually we start from the classical definition of λ-model à la Hindley-Longo (see [12]), with a further condition which forces the interpretation of $\mathsf{rec}\, x.\mathbf{t}$ as a fixed point of $\lambda x.\mathbf{t}$.

3.1. Definition. [models of $\lambda\mathsf{rec}$-calculus] A *model for the $\lambda\mathsf{rec}$-calculus consists of a triple $\langle \mathcal{D}, \cdot, [\![\;]\!]^{\mathcal{D}} \rangle$ such that $\mathcal{D}$ is a set, $\cdot : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$, $Env : Var \to \mathcal{D}$ and the interpretation function $[\![\;]\!]^{\mathcal{D}} : \Lambda \times Env \to \mathcal{D}$ satisfies:*

(i) $[\![x]\!]^{\mathcal{D}}_{\rho} = \rho(x)$;
(ii) $[\![\mathbf{t}\mathbf{u}]\!]^{\mathcal{D}}_{\rho} = [\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho} \cdot [\![\mathbf{u}]\!]^{\mathcal{D}}_{\rho}$;
(iii) $[\![\lambda x.\mathbf{t}]\!]^{\mathcal{D}}_{\rho} \cdot [\![\mathbf{u}]\!]^{\mathcal{D}}_{\rho} = [\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho[x:=[\![\mathbf{u}]\!]^{\mathcal{D}}_{\rho}]}$;
(iv) If $\rho(x) = \rho'(x)$ for all $x \in \mathrm{FV}(\mathbf{t})$, then $[\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho} = [\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho'}$;
(v) If $y \notin \mathrm{FV}(\mathbf{t})$, then $[\![\lambda x.\mathbf{t}]\!]^{\mathcal{D}}_{\rho} = [\![\lambda y.\mathbf{t}[x := y]]\!]^{\mathcal{D}}_{\rho}$;
(vi) If $\forall d \in \mathcal{D}.[\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho[x:=d]} = [\![\mathbf{u}]\!]^{\mathcal{D}}_{\rho[x:=d]}$, then $[\![\lambda x.\mathbf{t}]\!]^{\mathcal{D}}_{\rho} = [\![\lambda x.\mathbf{u}]\!]^{\mathcal{D}}_{\rho}$;
(vii) $[\![\mathsf{rec}\, x.\mathbf{t}]\!]^{\mathcal{D}}_{\rho} = [\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho[x:=[\![\mathsf{rec}\, x.\mathbf{t}]\!]^{\mathcal{D}}_{\rho}]}$.

$\langle \mathcal{D}, \cdot, [\![\;]\!]^{\mathcal{D}} \rangle$ is *extensional* if moreover when $x \notin \mathrm{FV}(\mathbf{t})$:

$$[\![\lambda x.\mathbf{t}x]\!]^{\mathcal{D}}_{\rho} = [\![\mathbf{t}]\!]^{\mathcal{D}}_{\rho}.$$

We now discuss how to build λ-models out of type theories. We start with the definition of *filter* for eitts. Then we show how to turn the space of filters into an applicative structure. We define continuous maps from the space of filters to the space of its continuous functions and vice versa. Since the composition of these maps is the identity we get standard λ-models (*filter models*). The interpretation of recursive terms is then obtained by using fixed point operators (we use the *minimal* one).

3.2. Definition. [Filters]

(i) A ∇-filter (or a filter over $\mathbb{T}^{\nabla}$) is a set $X \subseteq \mathbb{T}^{\nabla}$ such that:
- $\Omega \in X$;
- if $A \leq_{\nabla} B$ and $A \in X$, then $B \in X$;
- if $A, B \in X$, then $A \cap B \in X$;

(ii) $\mathcal{F}^{\nabla}$ denotes the set of ∇-filters over $\mathbb{T}^{\nabla}$;

(iii) if $X \subseteq \mathbb{T}^{\nabla}$, $\uparrow^{\nabla} X$ denotes the ∇-filter generated by X;

(iv) a ∇-filter is *principal* if it is of the shape $\uparrow^{\nabla}\{A\}$, for some type A. We shall denote $\uparrow^{\nabla}\{A\}$ simply by $\uparrow^{\nabla} A$.

It is well known that $\mathcal{F}^\nabla$ is an ω-algebraic lattice, whose poset of compact (or finite) elements is isomorphic to the reversed poset obtained by quotienting the preorder on $\mathbb{T}^\nabla$ by $\sim_\nabla$. That means that the compact elements are the filters of the form $\uparrow^\nabla A$ for some type A, the top element is $\mathbb{T}^\nabla$, and the bottom element is $\uparrow^\nabla\Omega$. Moreover the join of two filters is the filter induced by their union and the meet of two filters is their intersection, i.e.:

$$X \sqcup Y = \uparrow^\nabla (X \cup Y)$$
$$X \sqcap Y = X \cap Y.$$

The key property of $\mathcal{F}^\nabla$ is to be a reflexive object in the category of ω-algebraic complete lattices and Scott-continuous functions. This become clear by endowing the space of filters with a notion of application which induces continuous maps from $\mathcal{F}^\nabla$ to its function space $[\mathcal{F}^\nabla \to \mathcal{F}^\nabla]$ and vice versa.

3.3. DEFINITION. [Application]

(i) Application $\cdot : \mathcal{F}^\nabla \times \mathcal{F}^\nabla \to \mathcal{F}^\nabla$ is defined as

$$X \cdot Y = \{B \mid \exists A{\in}Y. A \to B{\in}X\}.$$

(ii) The continuous maps $\mathbb{F}^\nabla : \mathcal{F}^\nabla \to [\mathcal{F}^\nabla \to \mathcal{F}^\nabla]$ and $\mathbb{G}^\nabla : [\mathcal{F}^\nabla \to \mathcal{F}^\nabla] \to \mathcal{F}^\nabla$ are defined as:

$$\mathbb{F}^\nabla(X) = \lambda\!\!\lambda Y{\in}\mathcal{F}^\nabla. X \cdot Y;$$
$$\mathbb{G}^\nabla(f) = \uparrow^\nabla \{A \to B \mid B{\in}f(\uparrow^\nabla A)\}.$$

Notice that previous definition is sound, since it is easy to verify that $X \cdot Y$ is a ∇-filter.

As expected, $\mathbb{F}^\nabla$ and $\mathbb{G}^\nabla$ are inverse to each other: the proof is given in [2].

3.4. LEMMA.

$$\mathbb{F}^\nabla \circ \mathbb{G}^\nabla = id_{[\mathcal{F}^\nabla \to \mathcal{F}^\nabla]};$$
$$\mathbb{G}^\nabla \circ \mathbb{F}^\nabla = id^\nabla_{\mathcal{F}}.$$

We are now in position for interpreting the $\lambda\mathsf{rec}$-calculus. Let $\mathsf{Env}^\nabla_{\mathcal{F}}$ be the set of all mappings from the set of term variables to $\mathcal{F}^\nabla$ and ρ range over $\mathsf{Env}^\nabla_{\mathcal{F}}$. Let *fix* be the *minimal fixed point operator*, that is:

- $\forall X{\in}\mathcal{F}^\nabla.\ X \cdot (\mathit{fix}(X)) = \mathit{fix}(X)$;
- $\forall X, Y{\in}\mathcal{F}^\nabla. X \cdot Y = Y \ \Rightarrow \mathit{fix}(X) \subseteq Y$.

Via the maps $\mathbb{F}^\nabla$ and $\mathbb{G}^\nabla$, we define the *semantic interpretation* $[\![\ ]\!]^\nabla : \Lambda\mathsf{rec} \times$

$\mathsf{Env}^{\triangledown}_{\mathcal{F}} \to \mathcal{F}^{\triangledown}$ of $\lambda\mathsf{rec}$-terms:

$$\begin{aligned}
[\![x]\!]^{\triangledown}_{\rho} &= \rho(x);\\
[\![\mathbf{tu}]\!]^{\triangledown}_{\rho} &= \mathbb{F}^{\triangledown}([\![\mathbf{t}]\!]^{\triangledown}_{\rho})([\![\mathbf{u}]\!]^{\triangledown}_{\rho});\\
[\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho} &= \mathbb{G}^{\triangledown}(\lambda\!\!\lambda X{\in}\mathcal{F}^{\triangledown}.[\![\mathbf{t}]\!]^{\triangledown}_{\rho[X/x]});\\
[\![\mathsf{rec}\, x.\mathbf{t}]\!]^{\triangledown}_{\rho} &= \mathit{fix}([\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}).
\end{aligned}$$

As well-known, Lemma 3.4 implies that $\mathcal{F}^{\triangledown}$ induces an extensional λ-model, since it is a reflexive object in the cartesian closed category of ω-algebraic lattice, hence all the equations in Definition 3.1 hold, but for (vii), which follows immediately by definition of fixed point operator.

Next step is to relate the abstract notion of interpretation above, with the type assignment system. More specifically we want to prove that for any EITT $\Sigma^{\triangledown}$, $\lambda\mathsf{rec}$-term $\mathbf{t}$, and environment ρ, we have

$$[\![\mathbf{t}]\!]^{\triangledown}_{\rho} = \{A{\in}\mathbb{T}^{\triangledown} \mid \exists\Gamma \models \rho.\ \Gamma \vdash^{\triangledown} \mathbf{t} : A\},$$

where the notation $\Gamma \models \rho$ means that for $(x : B){\in}\Gamma$ one has that $B{\in}\rho(x)$.

In view of this we need a characterisation of *fix* as filter. Lemma 3.4 implies that every "higher order" space can be embedded in a canonical way in $\mathcal{F}^{\triangledown}$, by defining standard appropriate mappings via $\mathbb{F}^{\triangledown}$ and $\mathbb{G}^{\triangledown}$. For instance, in order to embed the space of *fix*, namely $[[\mathcal{F}^{\triangledown} \to \mathcal{F}^{\triangledown}] \to \mathcal{F}^{\triangledown}]$, in $\mathcal{F}^{\triangledown}$, we consider the pair of mappings $\mathbb{H}^{\triangledown} : \mathcal{F}^{\triangledown} \to [[\mathcal{F}^{\triangledown} \to \mathcal{F}^{\triangledown}] \to \mathcal{F}^{\triangledown}]$ and $\mathbb{K}^{\triangledown} : [[\mathcal{F}^{\triangledown} \to \mathcal{F}^{\triangledown}] \to \mathcal{F}^{\triangledown}] \to \mathcal{F}^{\triangledown}$ defined as follows:

$$\begin{aligned}
\mathbb{H}^{\triangledown}(X) &= \mathbb{F}^{\triangledown}(X) \circ \mathbb{G}^{\triangledown},\\
\mathbb{K}^{\triangledown}(\mathsf{H}) &= \mathbb{G}^{\triangledown} \circ \lambda\!\!\lambda X.(\mathsf{H} \circ \mathbb{F}^{\triangledown})(X).
\end{aligned}$$

It is easy to check that

$$(\natural) \quad \mathbb{H}^{\triangledown} \circ \mathbb{K}^{\triangledown} = id_{[[\mathcal{F}^{\triangledown}\to\mathcal{F}^{\triangledown}]\to\mathcal{F}^{\triangledown}]\to[[\mathcal{F}^{\triangledown}\to\mathcal{F}^{\triangledown}]\to\mathcal{F}^{\triangledown}]}.$$

We say that a filter X *represents* an operator $\mathsf{H}{\in}[[\mathcal{F}^{\triangledown} \to \mathcal{F}^{\triangledown}] \to \mathcal{F}^{\triangledown}]$ if $\mathbb{H}^{\triangledown}(X) = \mathsf{H}$. The equality ($\natural$) guarantees that each H is represented by $\mathbb{K}^{\triangledown}(\mathsf{H})$.

Given any EITT $\Sigma^{\triangledown}$, we now study a special filter, called $\Xi^{\triangledown}$, first introduced in [2], where $\Xi^{\triangledown}$ is proved to be a $\triangledown$-filter representing the *fix* operator (see Theorem 3.7).

3.5. DEFINITION. [Filter $\Xi^{\triangledown}$] Let $\Sigma^{\triangledown}$ an EITT. For all integers n, the sets $\Phi^{\triangledown}_n$ and the filters $\Psi^{\triangledown}_n$ are defined by mutual induction as follows:

$$\begin{aligned}
\Phi^{\triangledown}_0 &= \{\Omega\} & \Psi^{\triangledown}_0 &= \uparrow^{\triangledown}\Phi^{\triangledown}_0;\\
\Phi^{\triangledown}_{n+1} &= \{C \to A \mid \exists B.C \to B{\in}\Psi^{\triangledown}_n \ \&\ C \leq_{\triangledown} B \to A\} & \Psi^{\triangledown}_{n+1} &= \uparrow^{\triangledown}\Phi^{\triangledown}_{n+1}.
\end{aligned}$$

We define $\Xi^\triangledown = \bigcup_n \Psi_n^\triangledown$.

For instance $A \to \Omega \in \Phi_1^\triangledown$, $(\Omega \to A) \to A \in \Phi_2^\triangledown$, $(\Omega \to A_0) \cap (A_0 \to A_1) \to A_1 \in \Phi_3^\triangledown$, and $(\Omega \to A_0) \cap (A_0 \to A_1) \cap \ldots \cap (A_{n-1} \to A_n) \to A_n \in \Phi_{n+2}^\triangledown$ for all $A, A_0, \ldots, A_n$.

Now we give an useful lemma on $\Phi_n^\triangledown$, $\Psi_n^\triangledown$ which characterises the arrow types in $\Xi^\triangledown$.

3.6. LEMMA. (i) For all $n > 0$ we have $C \to A \in \Phi_n^\triangledown \;\Leftrightarrow\; C \to A \in \Psi_n^\triangledown$.
(ii) For all $n > 0$ we have

$$\begin{aligned} C \to A \in \Psi_n^\triangledown \Leftrightarrow \exists B_0, \ldots, B_n.\ C \leq_\triangledown \textstyle\bigcap_{0 \leq i \leq n-1} (B_i \to B_{i+1}) \\ \& \ B_0 \equiv \Omega \ \& \ B_n \equiv A. \end{aligned}$$

PROOF. (i) is proved in [2]. The proof of (ii) is by induction on n.

For $n = 1$ first we show:

$$(\sharp)\ C \to A \in \Psi_0^\triangledown \Leftrightarrow \Omega \leq_\triangledown A.$$

$$\begin{aligned} C \to A \in \Psi_0^\triangledown &\Leftrightarrow \Omega \leq_\triangledown C \to A && \text{by definition of } \Psi_0^\triangledown \\ &\Leftrightarrow \Omega \to \Omega \leq_\triangledown C \to A && \text{by axioms } (\Omega) \text{and } (\Omega\text{-}\eta) \\ &\Leftrightarrow \Omega \leq_\triangledown A && \text{by rule } (\eta) \text{ and Theorem 2.4.} \end{aligned}$$

Then we get:

$$\begin{aligned} C \to A \in \Psi_1^\triangledown &\Leftrightarrow C \to A \in \Phi_1^\triangledown && \text{by (i)} \\ &\Leftrightarrow \exists B.\ C \to B \in \Psi_0^\triangledown \ \& \ C \leq_\triangledown B \to A && \text{by definition of } \Phi_1^\triangledown \\ &\Leftrightarrow \exists B.\ \Omega \leq_\triangledown B \ \& \ C \leq_\triangledown B \to A && \text{by } (\sharp) \\ &\Leftrightarrow C \leq_\triangledown \Omega \to A && \text{by rule } (\eta) \text{ and Theorem 2.4.} \end{aligned}$$

For the induction step:

$$\begin{aligned} C \to A \in \Psi_{n+1}^\triangledown &\Leftrightarrow C \to A \in \Phi_{n+1}^\triangledown && \text{by (i)} \\ &\Leftrightarrow \exists B.\ C \to B \in \Psi_n^\triangledown \ \& \ C \leq_\triangledown B \to A && \text{by definition of } \Phi_{n+1}^\triangledown \\ &\Leftrightarrow \exists B, B_0, \ldots, B_n.\ C \leq_\triangledown \textstyle\bigcap_{0 \leq i \leq n-1} (B_i \to B_{i+1}) \\ &\qquad \& \ B_0 \equiv \Omega \ \& \ B_n \equiv B \ \& \ C \leq_\triangledown B \to A && \text{by induction} \\ &\Leftrightarrow \exists B_0, \ldots, B_n.\ B_0 \equiv \Omega \ \& && \text{by rules } (\mathit{mon}), \\ &\qquad C \leq_\triangledown (\textstyle\bigcap_{0 \leq i \leq n-1} (B_i \to B_{i+1})) \cap (B_n \to A) && (\mathit{trans}) \text{ and axiom } (\mathit{idem}). \end{aligned}$$

The key property of $\Xi^{\triangledown}$ proved in [2] is:

3.7. Theorem. The filter $\Xi^{\triangledown}$ represents the minimal fixed point operator, i.e. $\Xi^{\triangledown} = \mathbb{H}^{\triangledown}(\mathit{fix})$.

Having characterized *fix* through the filter $\Xi^{\triangledown}$ is fundamental for proving the next result.

3.8. Theorem.

$$[\![\mathbf{t}]\!]^{\triangledown}_{\rho} = \{A \in \mathbb{T}^{\triangledown} \mid \exists \Gamma \models \rho.\ \Gamma \vdash^{\triangledown} \mathbf{t} : A\}.$$

Proof. By induction on $\lambda\mathsf{rec}$-terms, using the Generation Theorem 2.9. All cases but that of rec-terms are proved in [2].

When considering terms $\mathsf{rec}\,x.\mathbf{t}$, by the characterization $[\![\mathsf{rec}\,x.\mathbf{t}]\!]^{\triangledown}_{\rho} = \Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}$, we have to show:

$$\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho} = \{A \in \mathbb{T}^{\triangledown} \mid \exists \Gamma \models \rho.\ \Gamma \vdash^{\triangledown} \mathsf{rec}\,x.\mathbf{t} : A\}.$$

We first show:

$$(\flat) \quad \Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho} = \{A \mid \exists n, B_0 \dots B_n, \Gamma \models \rho.\ B_0 \equiv \Omega \ \&\ B_n \equiv A \\ \&\ \Gamma, x : B_i \vdash^{\triangledown} \mathbf{t} : B_{i+1} (0 \leq i \leq n-1)\}.$$

Proof of $(\flat)$.

$$\begin{aligned}
\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho} &= \{A \mid \exists C \in [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}.\ C \to A \in \Xi^{\triangledown}\} \\
&\quad \text{by definition of application} \\
&= \{A \mid \exists C, \Gamma \models \rho.\ \Gamma \vdash^{\triangledown} \lambda x.\mathbf{t} : C \ \&\ C \to A \in \Xi^{\triangledown}\} \\
&\quad \text{by induction} \\
&= \{A \mid \exists C, \Gamma \models \rho.\ \Gamma \vdash^{\triangledown} \lambda x.\mathbf{t} : C \ \&\ \exists n, B_0 \dots B_n. \\
&\quad C \leq_{\triangledown} \textstyle\bigcap_{0 \leq i \leq n-1} (B_i \to B_{i+1}) \ \&\ B_0 \equiv \Omega \ \&\ B_n \equiv A\} \\
&\quad \text{by definition of } \Xi^{\triangledown} \text{ and Lemma 3.6(ii)} \\
&= \{A \mid \exists n, B_0 \dots B_n, \Gamma \models \rho.\ B_0 \equiv \Omega \ \&\ B_n \equiv A \\
&\quad \&\ \Gamma \vdash^{\triangledown} \lambda x.\mathbf{t} : B_i \to B_{i+1} (0 \leq i \leq n-1)\} \\
&\quad \text{by rule } (\leq_{\triangledown}) \\
&= \{A \mid \exists n, B_0 \dots B_n, \Gamma \models \rho.\ B_0 \equiv \Omega \ \&\ B_n \equiv A \\
&\quad \&\ \Gamma, x : B_i \vdash^{\triangledown} \mathbf{t} : B_{i+1} (0 \leq i \leq n-1)\} \\
&\quad \text{by the Generation Theorem (Theorem 2.9(iv)).}
\end{aligned}$$

Proof of $\subseteq$.

We check by induction on i that $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : B_i$ for $0 \leq i \leq n$. The basic step is trivial since $B_0 \equiv \Omega$.
For the induction step it suffices to apply rule $(\mathsf{rec}\,)$:

$$\frac{\Gamma, x{:}B_i \vdash^{\triangledown} \mathbf{t} : B_{i+1} \quad \Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : B_i}{\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : B_{i+1}}$$

We conclude $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$ since $B_n \equiv A$.

Proof of $\supseteq$.

This proof is by induction on $\nu(\mathcal{D})$ where $\mathcal{D}$ is a derivation of $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : A$. The basic step $\nu(\mathcal{D}) = 0$ is immediate.
For the induction step we get from the Generation Theorem (Theorem 2.9(v)) that there exist a non-empty set J, $\mathcal{D}_j$, E_j, D_j, such that:

(i) for any $i{\in}J$, $\mathcal{D}_j$ is a subderivation of $\mathcal{D}$ with conclusion $\Gamma \vdash^{\triangledown} \mathsf{rec}\, x.\mathbf{t} : D_j$;
(ii) $\forall j{\in}J.\Gamma, x{:}D_j \vdash^{\triangledown} \mathbf{t} : E_j$;
(iii) $\bigcap_{j\in J} E_j \leq_{\triangledown} A$;
(iv) $\Sigma_{j\in J}\nu(\mathcal{D}_j) + |J| = \nu(\mathcal{D})$.

From (i) and (iv) we get by induction $D_j{\in}\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}$ for all $i{\in}J$. This implies by $(\flat)$ that there exist $n_j, B^{(j)}_0 \ldots B^{(j)}_{n_j}$ such that $B^{(j)}_0 \equiv \Omega$, $B^{(j)}_{n_j} \equiv D_j$ and $\Gamma, x : B^{(j)}_i \vdash^{\triangledown} \mathbf{t} : B^{(j)}_{i+1} (0 \leq i \leq n_j - 1)$. Using (ii) and $(\flat)$ again we get $E_j{\in}\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}$, so we conclude $A{\in}\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}$ from (iii) being $\Xi^{\triangledown} \cdot [\![\lambda x.\mathbf{t}]\!]^{\triangledown}_{\rho}$ a $\triangledown$-filter.

We end by showing the following consistency result: given any *simple easy* terms $\mathbf{e}$, we can add to the calculus the countable amount of equations

$$(*_{\mathbf{e}}) \quad \forall \mathbf{t}{\in}\Lambda\mathsf{rec}\,.\ \mathbf{e}(\lambda x.\mathbf{t}) = \mathsf{rec}\, x.\mathbf{t}$$

and the resulting theory $\lambda\mathsf{rec} + (*_{\mathbf{e}})$ is still consistent.

We recall the definition of simple easy terms first done in [3]. A term is simple easy if we can force its interpretation to be extended exactly by an arbitrary principal filter. More precisely $\mathbf{e}$ is simple easy if, given an EITT $\Sigma^{\triangledown}$ and a type $Z{\in}\mathbb{T}^{\triangledown}$, we can extend in a conservative way $\Sigma^{\triangledown}$ to an EITT $\Sigma^{\triangledown'}$, so that $[\![\mathbf{e}]\!]^{\triangledown'} = (\uparrow^{\triangledown'} Z) \sqcup [\![\mathbf{e}]\!]^{\triangledown}$.

First we introduce *EITT maps*: an EITT map applied to an easy intersection type theory and to a type builds a new easy intersection type theory which is a conservative extension of the original one.

3.9. Definition. [EITT maps]

(i) Let $\Sigma^{\triangledown}$ and $\Sigma^{\triangledown'}$ two EITTs. We say that $\Sigma^{\triangledown'}$ is a *conservative extension* of

$\Sigma^{\triangledown}$ (notation $\Sigma^{\triangledown} \sqsubseteq \Sigma^{\triangledown'}$) if and only if $\mathbb{C}^{\triangledown} \subseteq \mathbb{C}^{\triangledown'}$ and for all $A, B \in \mathbb{T}^{\triangledown}$,

$$A \leq_{\triangledown} B \text{ if and only if } A \leq_{\triangledown'} B.$$

(ii) A *pointed* EITT is a pair $(\Sigma^{\triangledown}, Z)$ with $Z \in \mathbb{T}^{\triangledown}$.

(iii) An *EITT map* is a map $\mathsf{M} : \mathit{PEITT} \mapsto \mathit{EITT}$, such that for all $(\Sigma^{\triangledown}, Z)$

$$\Sigma^{\triangledown} \sqsubseteq \mathsf{M}(\Sigma^{\triangledown}, Z),$$

where EITT and PEITT denote respectively the class of EITTS and pointed EITTS.

3.10. DEFINITION. [Simple easy terms] An unsolvable term $\mathbf{e}$ is *simple easy* if there exists an EITT map $\mathsf{M}_{\mathbf{e}}$ such that for all pointed EITT $(\Sigma^{\triangledown}, Z)$,

$$\vdash^{\triangledown'} \mathbf{e} : B \text{ if and only if } \exists C \in \mathbb{T}^{\triangledown}. C \cap Z \leq_{\triangledown'} B \;\&\; \vdash^{\triangledown} \mathbf{e} : C,$$

where $\Sigma^{\triangledown'} = \mathsf{M}_{\mathbf{e}}(\Sigma^{\triangledown}, Z)$.

Define $\mathbf{I} \equiv \lambda x.x$, $\mathbf{W}_2 \equiv \lambda x.xx$, $\mathbf{W}_3 \equiv \lambda x.xxx$, and $\mathbf{R}_n$ inductively as $\mathbf{R}_0 = \mathbf{W}_2\mathbf{W}_2$, $\mathbf{R}_{n+1} = \mathbf{R}_n\mathbf{R}_n$. Examples of simple easy terms are $\mathbf{W}_2\mathbf{W}_2$, $\mathbf{W}_3\mathbf{W}_3\mathbf{I}$, and $\mathbf{R}_n$ for all n [3]. Lusin [15] gives further examples of simple easy terms.

The property of simple easy terms useful here is (for a proof see [2]):

3.11. THEOREM. Let $\mathbf{e}$ be a simple easy term. Then there exists a non-trivial filter model $\mathcal{F}^{\triangledown}$ such that the interpretation of $\mathbf{e}$ is the minimal fixed point operator.

We can then conclude:

3.12. THEOREM. Let $\mathbf{e}$ be a simple easy term. Then there exists a non-trivial filter model $\mathcal{F}^{\triangledown}$ such that the interpretation of $\mathbf{e}(\lambda x.\mathbf{t})$ and $\mathsf{rec}\, x.\mathbf{t}$ coincide for all $\mathbf{t} \in \Lambda\mathsf{rec}$.

3.13. COROLLARY. Let $\mathbf{e}$ be a simple easy term. Then the theory $\lambda\mathsf{rec} + (*_{\mathbf{e}})$ is consistent.

4. Conclusion

We end the paper justifying the decision of explicitly introducing terms $\mathsf{rec}\, x.\mathbf{t}$ and of *not* interpreting them by $[\![\mathbf{Y}(\lambda x.\mathbf{t})]\!]^{\triangledown}$ for some fixed point combinator $\mathbf{Y}$. A first reason is the smoothness of the type assignment system: deriving types for $\mathsf{rec}\, x.\mathbf{t}$ is very plain with respect to the cumbersome application of rules for deriving types to $\mathbf{Y}(\lambda x.\mathbf{t})$. As a second, and more important, reason, the possible interpretation of $\mathsf{rec}\, x.\mathbf{t}$ as minimal fixed point of $\lambda x.\mathbf{t}$ cannot be captured by any $\mathbf{Y}$. In fact, there is no fixed point combinator $\tilde{\mathbf{Y}}$ such that $[\![\tilde{\mathbf{Y}}]\!]^{\triangledown}$ represents the minimal fixed point operator *fix* in *each* filter model $\mathcal{F}^{\triangledown}$.

In fact, consider the filter model $\mathcal{F}^{Park}$ isomorphic to the Park λ-model $\mathcal{D}^{Park}$ of λ-calculus (see Honsell and Ronchi [13]). As proven in [13], for all closed λ-terms $\mathbf{t}$, $[\![\mathbf{t}]\!]^{Park}$ is above a certain compact element $\mathbf{c}$ different from the bottom element. In particular, for all fixed point combinators $\mathbf{Y}$, $[\![\mathbf{YI}]\!]^{Park}$ is above $\mathbf{c}$, where $\mathbf{I}$ is the identity combinator. Since $\mathit{fix}(\lambda\!\!\lambda X.X)$ is obviously the bottom element, we have that it is not possible that $\mathbb{H}^{Park}([\![\mathbf{Y}]\!]^{Park})$ represents *fix*, since

$$\begin{aligned}\mathbb{H}^{Park}([\![\mathbf{Y}]\!]^{Park})(\lambda\!\!\lambda X.X) &= (\mathbb{F}^{Park}([\![\mathbf{Y}]\!]^{Park}) \circ \mathbb{G}^{Park})(\lambda\!\!\lambda X.X)\\ &= \mathbb{F}^{Park}([\![\mathbf{Y}]\!]^{Park})(\mathbb{G}^{Park}(\lambda\!\!\lambda X.X))\\ &= [\![\mathbf{Y}]\!]^{Park} \cdot [\![\mathbf{I}]\!]^{Park}\\ &= [\![\mathbf{YI}]\!]^{Park}\\ &\sqsupseteq \mathbf{c}\end{aligned}$$

where we have used the fact that $[\![\mathbf{I}]\!]^{\nabla} = \mathbb{G}^{\nabla}(\lambda\!\!\lambda X.X)$ for all Σ^{∇}.

Acknowledgments

The authors are grateful to Furio Honsell for enlightening discussions on the subject of the present paper and to the referees for careful reading and useful suggestions which strongly improved the paper presentation.

References

[1] Fabio Alessi, Mariangiola Dezani-Ciancaglini, and Furio Honsell. Filter models and easy terms. In A.Restivo, S.Ronchi, and L.Roversi, editors, *Italian Conference on Theoretical Computer Science*, volume 2202 of *Lecture Notes in Computer Science*, pages 17–37, Berlin, 2001. Springer-Verlag.

[2] Fabio Alessi, Mariangiola Dezani-Ciancaglini, and Stefania Lusin. Intersection types and domain operators. *Theoret. Comput. Sci.*, 316(1–3):25–47, 2004.

[3] Fabio Alessi and Stefania Lusin. Simple easy terms. In S. van Bakel, editor, *Intersection Types and Related Systems*, volume 70 of *Electronic Lecture Notes in Theoretical Computer Science*. Elsevier, 2002.

[4] Jos Baeten and Benno Boerboom. ω can be anything it shouldn't be. *Indag.Math.*, 41:111–120, 1979.

[5] 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.

[6] Mario Coppo. An extended polymorphic type system for applicative languages. In *MFCS*, volume 88 of *LNCS*, pages 194–204. Springer, 1980.

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

[8] Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. Principal type schemes and λ-calculus semantics. In J.P.Seldin and J.R.Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 535–560. Academic Press, London, 1980.

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

[10] Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, and Silvia Likavec. Behavioural inverse limit models. *Theoret. Comput. Sci.*, 316(1–3):49–74, 2004.

[11] Martin Emms and Hans Leiss. Extending the type checker of standard ml by polymorphic recursion. *Theoret. Comput. Sci.*, 212(1–2):157–181, 1999.

[12] Roger Hindley and Giuseppe Longo. Lambda-calculus models and extensionality. *Z. Math. Logik Grundlag. Math.*, 26(4):289–310, 1980.

[13] Furio Honsell and Simona 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.

[14] Jan Kuper. On the Jacopini technique. *Inform. and Comput.*, 138:101–123, 1997.

[15] Stefania Lusin. *Intersection Types, Lambda Abstraction Algebras and Lambda Theories*. PhD thesis, Venice University, 2002. ftp://ftp.cs.unibo.it/pub/techreports/2003-07.pdf.

[16] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. The MIT Press, 1990.

[17] Alan Mycroft. Polymorphic type scheme and recursive definitions. Number 167 in LNCS, pages 217–228, 1984.

[18] David 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.

[19] Gordon Plotkin. Lcf considered as a programming language. *Theoret. Comput. Sci.*, 5:223–255, 1977.

[20] Garrel Pottinger. A type assignment for the strongly normalizable λ-terms. In J.P. Seldin and J.R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–577. Academic Press, London, 1980.

[21] Cécile Zylberajch. *Syntaxe et Semantique de la Facilité en Lambda-calcul*. PhD thesis, Université Paris VII, 1991.