Relating function spaces to resourced function spaces

Lidia Sánchez-Gil
Universidad Complutense de Madrid
lidiasg@mat.ucm.es

Mercedes Hidalgo-Herrero
Universidad Complutense de Madrid
mhidalgo@edu.ucm.es

Yolanda Ortega-Mallén
Universidad Complutense de Madrid
yolanda@sip.ucm.es

ABSTRACT

In order to prove the computational adequacy of the (operational) natural semantics for lazy evaluation with respect to a standard denotational semantics, Launchbury defines a resourced denotational semantics. This should be equivalent to the standard one when given infinite resources, but this fact cannot be so directly established, because each semantics produces values in a different domain. The values obtained by the standard semantics belong to the usual lifted function space $D = [D \to D]_\perp$, while those produced by the resourced semantics belong to $[C \to E]$ where E satisfies the equation $E = [[C \to E] \to [C \to E]]_\perp$ and C (the domain of resources) is a countable chain domain defined as the least solution of the domain equation $C = C_\perp$.

We propose a way to relate functional values in the standard lifted function space to functional values in the corresponding resourced function space. We first construct the initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$ following Abramsky's construction of the initial solution of $D = [D \to D]_\perp$. Then we define a "similarity" relation between values in the constructed domain and values in the standard lifted function space. This relation is inspired by Abramsky's applicative bisimulation.

Finally we prove the desired equivalence between the standard denotational semantics and the resourced semantics for the lazy λ-calculus.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*

Keywords

Domain theory, denotational semantics, λ-calculus.

1. MOTIVATION

There is a mismatch between the pure λ-calculus in the standard theory [4] and the practice of functional programming, or more precisely, the lazy functional languages. This situation has been pointed out clearly by Abramsky in his seminal paper [1], where he proposes a "lazy" theory based on the notion of *applicative transition systems* and introduces a suitable domain equation, i.e., $D = [D \to D]_\perp$, which has a nontrivial initial solution that constitutes a canonical model for the family of lazy languages. The construction of this initial solution is detailed in [2]. We adapt this construction to the case where resources are added to the computational model, so that we obtain the initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$, where C is a countable chain domain defined as the least solution of the domain equation $C = C_\perp$.

"Resourced" domains like E enable to define semantics that restrict the number of available resources. For instance, let us consider that only 60 seconds of CPU time are available and that each reduction needs one second; thus after 60 reductions the system gets blocked and the semantic value is $\perp$. Another example: Boudol, Curien and Lavatelli present in [5] a λ-calculus with resources to provide a control on the substitution process, so that a computation gets in deadlock when there are not enough resources to carry out all the substitutions. For this they use a different domain equation, namely $D = [\mathcal{M}(D) \to D]_\perp$, where $\mathcal{M}(D)$ stands for multisets of D.

The resourced domain E was used by Launchbury in [7] to define a (denotational) resourced semantics that was introduced to prove the computational adequacy of his (operational) natural semantics for lazy evaluation. In that resourced semantics each syntactic level (in the term to be evaluated) requires the consumption of one resource. This mimics the derivation process for the natural semantics, where one operational rule is applied to eliminate each syntactic level. Hence, denotations that consume a finite number of resources correspond to finite derivations. Nevertheless, the construction of the initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$ is not detailed in Launchbury's work, and we have found it described nowhere.

Our motivation for constructing an initial solution for $E = [[C \to E] \to [C \to E]]_\perp$ (let us call it *CValue*) is to define a correspondence between its elements and those of the lifted function space $D = [D \to D]_\perp$ (let us call it *Value*). This correspondence is necessary to prove that, provided an unbounded number of resources, the resourced semantics equals the standard denotational semantics. In [7] Launchbury simply states that both semantics compute the same values, but this is not exact because the values produced by the standard semantics belong to *Value*, while

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

those obtained by the resourced semantics belong to *CValue*. How can be related functional values belonging to different semantic domains? The obvious answer is to observe their behaviour when applying them to any argument; but arguments and results are functional values too, thus some kind of recursive definition is needed. We are inspired by the *applicative bisimulation* defined by Abramsky in [1] as the limit of a sequence of relations, each one allowing to observe the applicative behaviour of functions until some fixed depth. But in Abramsky's applicative bisimulation the "experiments" are common to both sides of the simulation — namely, they are syntactic terms— while in our case, they are again values belonging to different domains. Therefore, our version of the applicative bisimulation is in some way more general, and it can be useful for the study of simulations between distinct sets.

Summarizing, the contributions of this paper are: (1) the construction of the initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$, where C represents resources and satisfies the equation $C = C_\perp$, (2) the definition of a "similarity" relation between values in the constructed domain and values in the standard lifted function space $D = [D \to D]_\perp$, and (3) the proof of the equivalence between a standard denotational semantics and a resourced semantics for the lazy λ-calculus.

These results allow us to obtain a proof for the computational adequacy of Launchbury's natural semantics. As we have pointed out above, Launchbury just assumes that values obtained from both denotational semantics can be directly related, but the adequacy proof is not detailed. To the best of our knowledge, no complete and detailed proof of this adequacy has been published before. When trying to reproduce it we have discovered that it is not a straightforward proof, and that several subtle difficulties arise.

The adequacy result is of interest since Launchbury's natural semantics has been often cited and has inspired many other works as well as several extensions of his semantics, where the corresponding adequacy proofs have been obtained by adapting Launchbury's proof scheme, e.g., [14, 3, 15, 9, 11]. Moreover, the resourced function space and the similarity relation defined here can be useful in other contexts.

The paper is organized as follows: In Section 2 we define the resourced function space and we detail the construction of the initial solution of the corresponding domain equation. In Section 3 we define a similarity relation between the values of the standard function space and those of the resourced function space. In Section 4 we describe a standard and a resourced denotational semantics for a lazy λ-calculus, and we prove their equivalence. The last section is devoted to the conclusions and the outline of future work.

2. A RESOURCED FUNCTION SPACE

A *partially ordered set* $(D, \sqsubseteq_D)$ is a *directed-complete partial order* (dcpo) if every directed subset has a supremum. A *complete partial order* (cpo) is a dcpo with a least element. For simplicity a cpo is usually represented by just its set.

A function $f : D \to D'$ between cpos D and D' is continuous if it maps directed sets to directed sets while preserving their suprema.

Following Scott's domain theory [6], the *function space* $[D \to D']$, being $(D, \sqsubseteq_D)$ and $(D', \sqsubseteq_{D'})$ cpos, is defined as $[D \to D'] \stackrel{\text{def}}{=} \{f \mid f : D \to D' \text{ is continuous}\}$, where the order is $f \sqsubseteq_{[D\to D']} g \stackrel{\text{def}}{=} \forall d \in D. f(d) \sqsubseteq_{D'} g(d)$.

The *lifted* construction of a cpo $(D, \sqsubseteq_D)$ is defined as:

$$\begin{array}{lll} D_\perp & \stackrel{\text{def}}{=} & \{\perp\} \cup \{\lfloor d \rfloor \mid d \in D\}, \\ x \sqsubseteq_{D_\perp} y & \stackrel{\text{def}}{=} & x = \perp \vee (x = \lfloor d \rfloor \wedge y = \lfloor d' \rfloor \wedge d \sqsubseteq_D d'). \end{array}$$

The function $\lfloor - \rfloor$ verifies that $\lfloor d \rfloor = \lfloor d' \rfloor \Rightarrow d = d'$ and $\perp \neq \lfloor d \rfloor$ for all $d, d' \in D$.

Two functions, $\mathsf{up}_D : D \to D_\perp$ and $\mathsf{dn}_D : D_\perp \to D$, relate the cpo D to its lifted version $D_\perp$ and viceversa:

$$\begin{array}{lll} \mathsf{up}_D(d) & \stackrel{\text{def}}{=} & \lfloor d \rfloor, \\ \\ \mathsf{dn}_D(\perp) & \stackrel{\text{def}}{=} & \perp_D, \\ \mathsf{dn}_D(\lfloor d \rfloor) & \stackrel{\text{def}}{=} & d. \end{array}$$

They verify that $\mathsf{dn}_D \circ \mathsf{up}_D = id_D$ and $id_{D_\perp} \sqsubseteq \mathsf{up}_D \circ \mathsf{dn}_D$.

There are different ways of realising $\lfloor - \rfloor$, all of them leading to isomorphic constructions. In the following we will consider that $\lfloor d \rfloor = d$.

Let $f : D \to D'$, it can be extended to $f_\perp : D_\perp \to D'_\perp$:

$$\begin{array}{lll} f_\perp(\perp) & \stackrel{\text{def}}{=} & \perp \text{ , and} \\ f_\perp(\mathsf{up}_D(d)) & \stackrel{\text{def}}{=} & \mathsf{up}_{D'}(f(d)) \text{ for } d \in D. \end{array}$$

The construction of the initial solution of the domain equation $D = [D \to D]_\perp$ is detailed in [2]. The objective of the present section is to adapt this construction to obtain the initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$, being C the least solution of the domain equation $C = C_\perp$. The elements of C are represented as $\perp$, $S(\perp)$, $S^2(\perp) = S(S(\perp))$, ..., $S^n(\perp)$, ..., being $S^\infty(\perp)$ its limit element, where S stands for a successor-like function. In the following, to simplify, we abbreviate $S^\infty(\perp)$ to S^∞.

The initial solution for the domain equation $E = [[C \to E] \to [C \to E]]_\perp$ is constructed by finite approximations. Each pair injection/projection between successive approximations constitutes an embedding.

2.1 Embeddings

Let D and D' be cpos. An *embedding* of D into D' is a pair of continuous maps $\langle i, j \rangle$ such that $D \stackrel{i}{\rightarrowtail} D' \stackrel{j}{\twoheadrightarrow} D$, and $i \circ j \sqsubseteq id_{D'}$ and $j \circ i = id_D$, where $\rightarrowtail$ and $\twoheadrightarrow$ stand for an injection and a projection respectively.

Given an embedding of D into D', it is possible to build other embeddings between cpos constructed from D and D'.

Consider the composition application over cpos X, Y and Z with the following signature:

$$\circ : [Y \to Z] \times [X \to Y] \to [X \to Z].$$

Then, for any $f \in [Y \to Z]$ and $g \in [X \to Y]$, we can define the function sections:

$$\begin{array}{l} (f \circ) : [X \to Y] \to [X \to Z] \text{ and} \\ (\circ\, g) : [Y \to Z] \to [X \to Z], \end{array}$$

so that $(f \circ)(h) \stackrel{\text{def}}{=} (f \circ h)$ and $(\circ\, g)(h) \stackrel{\text{def}}{=} (h \circ g)$ for a function h of the appropiate type in each case.

LEMMA 1. *Let $\langle i, j \rangle$ be an embedding of D into D'. For any cpo X, $\langle (i \circ), (j \circ) \rangle$ is an embedding of $[X \to D]$ into $[X \to D']$.*

PROOF. We have to show that $(j \circ) \circ (i \circ) = id_{[X\to D]}$ and $(i \circ) \circ (j \circ) \sqsubseteq id_{[X\to D']}$.

Let $f \in [X \to D]$, we have $(j \circ) \circ (i \circ)(f) = (j \circ) \circ (i \circ f) = (j \circ i) \circ f = id_D \circ f = f$, by associativity of $\circ$.

Similarly, if $g \in [X \to D']$ then $(i \circ) \circ (j \circ)(g) = (i \circ) \circ (j \circ g) = (i \circ j) \circ g \sqsubseteq id_{D'} \circ g = g$. $\square$

As a consequence, from an embedding of D into D', an embedding between the resourced domains $[C \to D]$ and $[C \to D']$ can be built by choosing X as the countable chain domain C defined above.

COROLLARY 2. *Let $\langle i, j\rangle$ be an embedding of D into D'. Then $\langle i^C, j^C\rangle$ is an embedding of $[C \to D]$ into $[C \to D']$, which is defined as $i^C(a)(c) \stackrel{def}{=} i(a(c))$ and $j^C(b)(c) \stackrel{def}{=} j(b(c))$, where $a \in [C \to D]$, $b \in [C \to D']$ and $c \in C$.*

PROOF. Notice that $i^C = (i \circ)$ and $j^C = (j \circ)$; then we use Lemma 1. □

Given an embedding between two cpos, an embedding between their lifted domains can be easily obtained.

LEMMA 3. *Let $\langle i, j\rangle$ be an embedding of D into D'. The pair $\langle i_\perp, j_\perp\rangle$ is an embedding of $D_\perp$ into $D'_\perp$.*

PROOF. Trivial from the definition of $i_\perp$ and $j_\perp$. □

Next we define how to build an embedding between function spaces from an embedding between their ground cpos.

LEMMA 4. *Let $\langle i, j\rangle$ be an embedding of D into D'. Then $\langle i^\to, j^\to\rangle$ is an embedding of $[D \to D]$ into $[D' \to D']$, where:*

$$\begin{array}{rcl} i^\to & \stackrel{def}{=} & (i \circ) \circ (\circ\ j), \text{ and} \\ j^\to & \stackrel{def}{=} & (j \circ) \circ (\circ\ i). \end{array}$$

PROOF. Since $\langle i, j\rangle$ is an embedding of D into D', it is verified that $j \circ i = id_D$ and $i \circ j \sqsubseteq id_{D'}$. We have to check that $j^\to \circ i^\to = id_{[D\to D]}$ and $i^\to \circ j^\to \sqsubseteq id_{[D'\to D']}$. We use again the associativity of $\circ$.

Let $f \in [D \to D]$ and $g \in [D' \to D']$:

$$\begin{array}{rcl} (i^\to \circ\ j^\to)(g) & = & ((i \circ) \circ (\circ\ j)) \circ ((j \circ) \circ (\circ\ i))(g) \\ & = & ((i \circ) \circ (\circ\ j))(j \circ g \circ i) \\ & = & i \circ (j \circ g \circ i) \circ j \\ & = & (i \circ j) \circ g \circ (i \circ j) \\ & \sqsubseteq & id'_D \circ g \circ id'_D \\ & = & g. \end{array}$$

Similarly, it is proved that $(j^\to \circ\ i^\to)(f) = f$. □

Now we combine the two previous constructions to obtain an embedding between lifted function spaces.

COROLLARY 5. *Let $\langle i, j\rangle$ be an embedding of D into D'. The pair $\langle i^*, j^*\rangle$, which is defined as $i^* \stackrel{def}{=} (i^\to)_\perp$ and $j^* \stackrel{def}{=} (j^\to)_\perp$, is an embedding of $[D \to D]_\perp$ into $[D' \to D']_\perp$.*

PROOF. Is a consequence of Lemmas 3 and 4. □

This corollary is equivalent to Lemma 4.1.1. given by Abramsky and Ong in [2]; although their definition of $\langle i^*, j^*\rangle$ is more cumbersome because the lifting and function space constructions are combined in a unique step.

These results also can be easily obtained by considering embeddings as Galois connections [8].

We are now ready to define an embedding of $[[C \to D] \to [C \to D]]_\perp$ into $[[C \to D'] \to [C \to D']]_\perp$. We start with an embedding $\langle i, j\rangle$ of D into D'; by Corollary 2 we obtain $\langle i^C, j^C\rangle$, an embedding of $[C \to D]$ into $[C \to D']$. Finally we get an embedding $\langle i^{C*}, j^{C*}\rangle$ of $[[C \to D] \to [C \to D]]_\perp$ into $[[C \to D'] \to [C \to D']]_\perp$ by applying Corollary 5 (see Figure 1).

The embedding $\langle i^{C*}, j^{C*}\rangle$ is used in the next subsection to construct the initial solution of $E = [[C \to E] \to [C \to E]]_\perp$.

2.2 Construction of the initial solution

We represent the finite aproximations of E by $\{E_n\}_{n\in\mathbb{N}}$, defined inductively as $E_0 \stackrel{\text{def}}{=} \{\perp_{E_0}\}$ and $E_{n+1} \stackrel{\text{def}}{=} [[C \to E_n] \to [C \to E_n]]_\perp$. Intuitively, E_n contains the (resourced) functions that can be applied until level n at most. Hence, E_0 contains only the bottom value, indicating that it cannot be applied. While E_1 has two elements, i. e., $E_1 = \{\perp_{E_1}, e_1\}$ where e_1 is the function that maps a_0 into a_0, being $A_0 = [C \to E_0] = \{a_0\}$ such that $\forall n \in \mathbb{N}^\infty (= \mathbb{N} \cup \{\infty\}).a_0(S^n(\perp)) = \perp_{E_0}$. Therefore, e_1 can be applied in a first level (to $a_0 \in A_0$) but the resulting value (a_0) instantiated with any amount of resources produces $\perp_{E_0}$, which cannot be applied.

Let $\langle i_0, j_0\rangle$ be an embedding of E_0 into E_1, where i_0 and j_0 are defined as follows:

$$\begin{array}{rcccc} i_0 & : & E_0 & \longrightarrow & E_1 \\ & & \perp_{E_0} & \mapsto & \perp_{E_1} \end{array} \qquad \begin{array}{rcccc} j_0 & : & E_1 & \longrightarrow & E_0 \\ & & \perp_{E_1} & \mapsto & \perp_{E_0} \\ & & e_1 & \mapsto & \perp_{E_0} \end{array}$$

The embeddings of E_{n+1} into E_{n+2}, i. e. $E_{n+1} \overset{i_{n+1}}{\rightarrowtail} E_{n+2} \overset{j_{n+1}}{\twoheadrightarrow} E_{n+1}$, are defined as $\langle i_{n+1}, j_{n+1}\rangle \stackrel{\text{def}}{=} \langle i_n^{C*}, j_n^{C*}\rangle$, that is,

$$\begin{array}{rcl} i_{n+1}(e_{n+1}) = i_n^{C*}(e_{n+1}) & = & (i_n^{C\to})_\perp(e_{n+1}) \\ & = & \begin{cases} \perp_{E_{n+2}} & \text{if } e_{n+1} = \perp_{E_{n+1}} \\ i_n^C \circ\ e_{n+1} \circ\ j_n^C & \text{if } e_{n+1} \neq \perp_{E_{n+1}} \end{cases} \end{array}$$

$$\begin{array}{rcl} j_{n+1}(e_{n+2}) = j_n^{C*}(e_{n+2}) & = & (j_n^{C\to})_\perp(e_{n+2}) \\ & = & \begin{cases} \perp_{E_{n+1}} & \text{if } e_{n+2} = \perp_{E_{n+2}} \\ j_n^C \circ\ e_{n+2} \circ\ i_n^C & \text{if } e_{n+2} \neq \perp_{E_{n+2}} \end{cases} \end{array}$$

$\langle E_n, j_n\rangle_{n\in\mathbb{N}}$ forms an inverse system of cpos and we take its inverse limit [13] as E, i.e.,

$$\begin{array}{l} E = \lim_{\leftarrow}\langle E_n, j_n\rangle \\ \quad = \{\langle e_n : n \in \mathbb{N}\rangle \mid e_n \in E_n \wedge j_n(e_{n+1}) = e_n\} \subseteq \prod_{n\in\mathbb{N}} E_n. \end{array}$$

where the tuple $\langle e_n : n \in \mathbb{N}\rangle = \langle e_0, e_1, \ldots, e_n, \ldots\rangle$ represents an element $e \in E$ by its approximations in each layer.

The embeddings of E_n into E_{n+1} can be generalized to relate E_m to E_n, for any $n, m \in \mathbb{N}^\infty$.

We define the functions $\phi^E_{m,n} : E_m \to E_n$ as follows:

$$\begin{array}{ll} m = n & \phi^E_{n,n} \stackrel{\text{def}}{=} id_{E_n}, \\ m > n & \phi^E_{m,n} \stackrel{\text{def}}{=} \phi^E_{m-1,n} \circ\ j_{m-1}, \\ m < n & \phi^E_{m,n} \stackrel{\text{def}}{=} i_{n-1} \circ\ \phi^E_{m,n-1}. \end{array}$$

The *n-projection* $\phi^E_{\infty,n} : E \to E_n$ is defined as the limit of the projections $\phi^E_{m,n}$. For simplicity we write ψ^E_n for $\phi^E_{\infty,n}$. Similarly, ϕ^E_n represents the *n-injection* $\phi^E_{n,\infty} : E_n \to E$. By construction $\langle \phi^E_n, \psi^E_n\rangle$ forms an embedding of E_n into E.

Also, we view each E_n as a subset of E, that is, we identify $\phi^E_n(x)$ with x, where $x \in E_n$; and for $e \in E$, $\psi^E_n(e) = e_n \in E_n$. Thus, $E = \bigcup_{n\in\mathbb{N}} E_n$.

To illustrate our construction, we generate the three first approximations of E, i.e., E_0, E_1 and E_2, and we look into the corresponding embeddings.

As explained above, E_0 is the one-point domain represented by $\{\perp_{E_0}\}$, and $E_1 = \{\perp_{E_1}, e_1\}$, where $e_1(a_0) = a_0$ and $\{a_0\} = A_0 = [C \to E_0]$ is such that $a_0(S^n(\perp)) = \perp_{E_0}$ for all $n \in \mathbb{N}^\infty$. Now $E_2 = [A_1 \to A_1]_\perp$, where $A_1 = [C \to E_1]$ (see Figure 2). The elements of A_1 are functions $a_{1,n}$ ($n \in \mathbb{N}$)

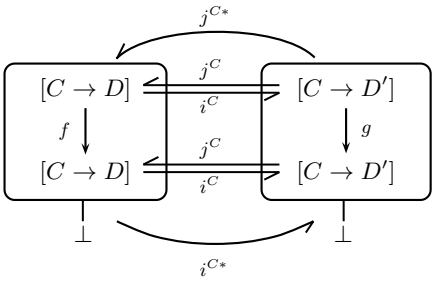

Figure 1: Embeddings on resourced domains

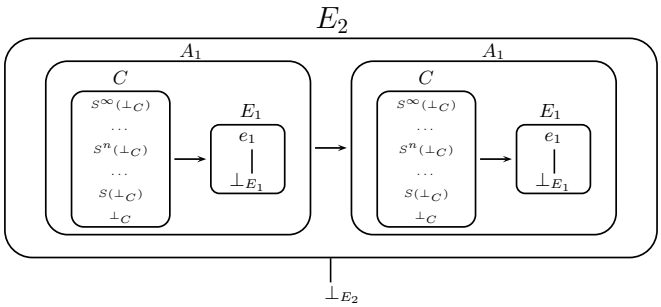

Figure 2: The domain E_2

such that:

$$a_{1,n}(S^k(\perp)) = \begin{cases} \perp_{E_1} & \text{if } k < n \\ e_1 & \text{if } k \geq n \end{cases}$$

We define $a_{1,\infty}$ as the function verifying that forall $n \in \mathbb{N}^\infty$ $a_{1,\infty}(S^n(\perp)) = \perp_{E_1}$, i.e., this is the least defined function in A_1. Notice that the most defined one is $a_{1,0}$, the constant function with $a_{1,0}(S^n(\perp)) = e_1$ forall $n \in \mathbb{N}^\infty$. Hence, we have an ordering in A_1 where $a_{1,\infty} \sqsubseteq a_{1,m} \sqsubseteq a_{1,n} \sqsubseteq a_{1,0}$ for all $m, n \in \mathbb{N}$ such that $n \leq m$.

Since $A_1 = \{a_{1,n} \mid n \in \mathbb{N}^\infty\}$, any increasing ω-chain in A_1 has as l.u.b. $a_{1,k}$ for some $k \in \mathbb{N}$, except for the constant chain $\{x_n = a_{1,\infty}\}_{n\in\mathbb{N}}$. Hence, we can characterize the continuous functions $e : A_1 \to A_1$ as those that satisfy the following property:

$$n \leq m \Rightarrow l \leq k, \text{ for } e(a_{1,m}) = a_{1,k} \text{ and } e(a_{1,n}) = a_{1,l} \quad (1)$$

Summarizing,

$$E_2 = \{\perp_{E_2}\} \cup \{e : \{a_{1,n}\}_{n\in\mathbb{N}^\infty} \to \{a_{1,n}\}_{n\in\mathbb{N}^\infty} \mid e \text{ sat.}(1)\}.$$

The representation of the embeddings $\langle i_0, j_0\rangle$ of E_0 into E_1 and $\langle i_1, j_1\rangle$ of E_1 into E_2 is shown in Figure 3. For E_2 we only highlight the constant functions $e_{2,\infty}$ and $e_{2,0}$ representing the two extremes of $[A_1 \to A_1]$, i.e., for all $n \in \mathbb{N}^\infty$

$$e_{2,\infty}(a_{1,n}) = a_{1,\infty} \quad \text{and} \quad e_{2,0}(a_{1,n}) = a_{1,0}.$$

2.3 Application operations

Application operations are defined in each domain E_n as $\mathsf{Ap}^\perp_{E_n} : E_{n+1} \times A_n \times C \to E_n$,

$$\mathsf{Ap}^\perp_{E_n}(e_{n+1}, a_n, c) \stackrel{\text{def}}{=} \begin{cases} \perp_{E_n} & \text{if } e_{n+1} = \perp_{E_{n+1}} \\ e_{n+1}(a_n)(c) & \text{if } e_{n+1} \neq \perp_{E_{n+1}} \end{cases}$$

ψ^E_n denotes the n-projection defined in Section 2.2 for the domain E. From the embedding $\langle \phi^E_n, \psi^E_n\rangle$ of E_n into E and using Corollary 2, we obtain the embedding $\langle (\phi^E_n)^C, (\psi^E_n)^C\rangle$ of $[C \to E_n]$ into $[C \to E]$, so that $\psi^{[C\to E]}_n$ stands for $(\psi^E_n)^C$. Application operation in E is $\mathsf{AP}^\perp_E : E \times [C \to E] \times C \to E$,

$$\mathsf{AP}^\perp_E(e, a, c) \stackrel{\text{def}}{=} \bigsqcup_{n\in\mathbb{N}} \mathsf{Ap}^\perp_{E_n}(\psi^E_{n+1}(e), \psi^{[C\to E]}_n(a), c).$$

In the following, we use $e\,(a)\,(c)$ for both $\mathsf{Ap}^\perp_{E_n}(e, a, c)$ and $\mathsf{AP}^\perp_E(e, a, c)$; from the context it should be clear which is the correct one. Notice that

$$\psi^E_n(e(a)(c)) = \psi^E_{n+1}(e)(\psi^{[C\to E]}_n(a))(c). \quad (2)$$

Analogously, from the definition of the application operation in the standard lifted function space D described in [2], where D is the initial solution of $D = [D \to D]_\perp$ and this initial solution can be described as an inverse limit similarly to E, it can be observed that

$$\psi^D_n(d(d')) = \psi^D_{n+1}(d)(\psi^D_n(d')). \quad (3)$$

3. SIMILARITY

In this section we define a relation between the standard lifted function domain $D = [D \to D]_\perp$ and the resourced domain $E = [[C \to E] \to [C \to E]]_\perp$ constructed in the previous section. Our relation is inspired by the *applicative bisimulation* defined by Abramsky in [1], so that functions $d \in D$ and $e \in E$ are considered to be *similar* if d and e have a *similar* applicative behaviour when infinite resources are available for e, i.e., they produce *similar* values when applied to *similar* arguments.

To formalize this circular definition we resort to a layered construction, that is, we inductively define a sequence of relations $\{ \lhd\rhd_n \subseteq D_n \times E_n\}_{n\in\mathbb{N}}$.

DEFINITION 1 (n-SIMILARITY). *A family of n-similarity relations between elements of D_n and E_n, for $n \in \mathbb{N}$, is defined as follows:*

- *0-similarity, $\lhd\rhd_0 : \perp_{D_0} \lhd\rhd_0 \perp_{E_0}$.*
- *n+1-similarity, $\lhd\rhd_{n+1}$, is the least relation in $D_{n+1} \times E_{n+1}$ that verifies:*
  (1) $\perp_{D_{n+1}} \lhd\rhd_{n+1} \perp_{E_{n+1}}$ and,
  (2) let $d \in D_{n+1}$ and $e \in E_{n+1}$ such that $d \neq \perp_{D_{n+1}}$ and $e \neq \perp_{E_{n+1}}$, then $d \lhd\rhd_{n+1} e$ if for any $d' \in D_n$ and for any $a' \in A_n$, $d' \lhd\rhd_n a'(S^\infty) \Rightarrow d(d') \lhd\rhd_n e(a')(S^\infty)$.

For a better understanding of this definition, we show the three first layers (Figure 4):

• 0-*similarity* ($\lhd\rhd_0$)*:* $D_0 = \{\perp_{D_0}\}$ and $E_0 = \{\perp_{E_0}\}$, so that the relation reduces to $\perp_{D_0} \lhd\rhd_0 \perp_{E_0}$.

• 1-*similarity* ($\lhd\rhd_1$)*:* $D_1 = \{\perp_{D_1}, d_1\}$, where $d_1 \in [D_0 \to D_0]$ is the function mapping $\perp_{D_0}$ into $\perp_{D_0}$, and recall that $E_1 = \{\perp_{E_1}, e_1\}$ with $e_1(a_0) = a_0$. $\perp_{D_1} \lhd\rhd_1 \perp_{E_1}$, by definition. Now let $d' \in D_0$ and $a' \in A_0 = \{a_0 \mid \forall n \in \mathbb{N}^\infty.\ a_0(S^n(\perp)) = \perp_{E_0}\}$, it must be that $d' = \perp_{D_0}$ and $a' = a_0$. Hence, we have that $\perp_{D_0} = d' \lhd\rhd_0 a'(S^\infty) = \perp_{E_0}$. Moreover, $d_1(d') = \perp_{D_0} \lhd\rhd_0 \perp_{E_0} = a_0(S^\infty) = e_1(a_0)(S^\infty) = e_1(a')(S^\infty)$. Thus, $d_1 \lhd\rhd_1 e_1$.

• 2-*similarity* ($\lhd\rhd_2$)*:* $D_2 = \{\perp_{D_2}, d_{2,1}, d_{2,2}, d_{2,3}\}$, where the functions $d_{2,1}, d_{2,2}$ and $d_{2,3}$ are defined as follows:

$$\begin{array}{lll} d_{2,1}(\perp_{D_1}) = \perp_{D_1} & d_{2,2}(\perp_{D_1}) = \perp_{D_1} & d_{2,3}(\perp_{D_1}) = d_1 \\ d_{2,1}(d_1) = \perp_{D_1} & d_{2,2}(d_1) = d_1 & d_{2,3}(d_1) = d_1 \end{array}$$

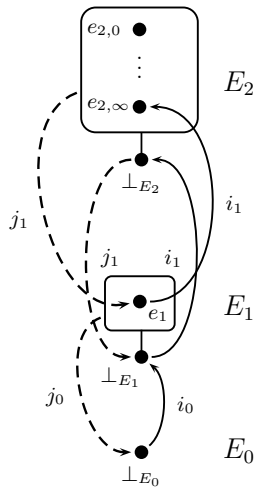

Figure 3: First steps of the construction of E

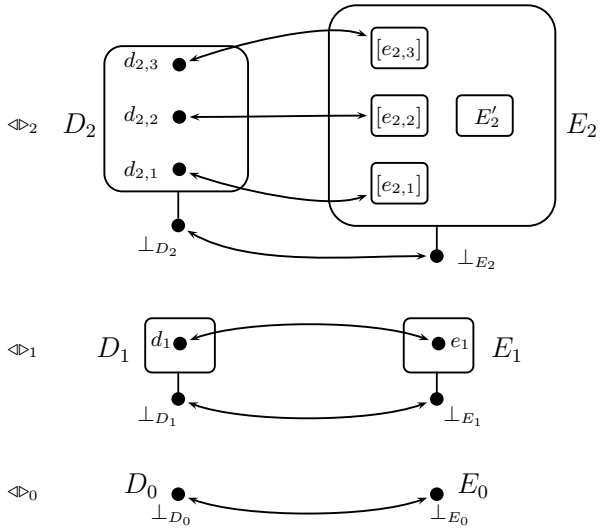

Figure 4: 0,1,2-similarity

We have shown in Subsection 2.2 that
$E_2 = \{\perp_{E_2}\} \cup \{e : \{a_{1,n}\}_{n\in\mathbb{N}^\infty} \to \{a_{1,n}\}_{n\in\mathbb{N}^\infty} \mid e \text{ sat. } (1)\}$.
By definition, $\perp_{D_2} \lhd\rhd_2 \perp_{E_2}$.
Let $d' \in D_1$ and $a' \in A_1 = [C \to E_1]$ such that $d' \lhd\rhd_1 a'(S^\infty)$. Consequently, either $d' = \perp_{D_1}$ and $a' = a_{1,\infty}$ (and hence $a'(S^\infty) = \perp_{E_1}$), or $d' = d_1$ and $a' = a_{1,k}$ for some $k \in \mathbb{N}$ (notice that $\forall n \in \mathbb{N} \,.\, a_{1,n}(S^\infty) = e_1$). Therefore, the functions in A_1 can be partitioned into two classes: $[a_{1,\infty}]$ is the class of functions returning $\perp_{E_1}$ when applied to S^∞; whereas the functions in $[a_{1,0}]$ return e_1 when infinite resources are provided, i.e., $[a_{1,0}] = \{a_{1,n} \mid a_{1,n}(S^\infty) = e_1\}$. Notice that the first class contains a unique element, that is, $[a_{1,\infty}] = \{a_{1,\infty}\}$. But it could be proved that 2-similarity requires that the classes in A_1 are preserved, i.e., for any $e \in E_2$ there exists $d \in D_2$ such that $d \lhd\rhd_2 e$ only if

$$\forall a, a' \in A_1.(a(S^\infty) = a'(S^\infty) \Rightarrow e(a)(S^\infty) = e(a')(S^\infty)).$$

Therefore, $E_2 = \{\perp_{E_2}\} \cup [e_{2,1}] \cup [e_{2,2}] \cup [e_{2,3}] \cup E_2'$. where

$$\begin{aligned}
[e_{2,1}] &= \{e \in E_2 \mid \forall a \in A_1 \,.\, e(a) \in [a_{1,\infty}]\},\\
[e_{2,2}] &= \{e \in E_2 \mid \forall a \in A_1 \,.\, (a \in [a_{1,\infty}] \Rightarrow e(a) \in [a_{1,\infty}])\\
&\qquad\qquad \wedge (a \in [a_{1,0}] \Rightarrow e(a) \in [a_{1,0}])\},\\
[e_{2,3}] &= \{e \in E_2 \mid \forall a \in A_1 \,.\, e(a) \in [a_{1,0}]\}.
\end{aligned}$$

Hence, we have that $\perp_{D_2} \lhd\rhd_2 \perp_{E_2}$, $\forall e \in [e_{2,1}].d_{2,1} \lhd\rhd_2 e$, $\forall e \in [e_{2,2}].d_{2,2} \lhd\rhd_2 e$, $\forall e \in [e_{2,3}].d_{2,3} \lhd\rhd_2 e$, and E_2' contains the elements of E_2 for which there is no similar element in D_2. This is graphically represented in Figure 4.

Next we prove some useful properties of the n-similarity, e.g., that it preserves undefinedness:

LEMMA 6. *Let $n \in \mathbb{N}$, $d \in D_n$ and $e \in E_n$. If $d \lhd\rhd_n e$ then either $(d = \perp_{D_n} \wedge e = \perp_{E_n})$ or $(d \neq \perp_{D_n} \wedge e \neq \perp_{E_n})$.*

PROOF. Trivial by the definition of $\lhd\rhd_n$. □

In the following lemma $\langle i_n^D, j_n^D\rangle$ stands for the embedding of D_n into D_{n+1} [1], $\langle i_n^E, j_n^E\rangle$ for the embedding of E_n into E_{n+1}, and $\langle i_n^C, j_n^C\rangle$ for the embedding of $[C \to E_n]$ into $[C \to E_{n+1}]$. The lemma states that the injections and projections of these embeddings preserve the similarity relation.

LEMMA 7. *Let $n \in \mathbb{N}$, $d \in D_{n+1}$, $e \in E_{n+1}$, $a \in A_{n+1}$, $d' \in D_n$, $e' \in E_n$ and $a' \in A_n$:*

1. $d \lhd\rhd_{n+1} e \Rightarrow j_n^D(d) \lhd\rhd_n j_n^E(e)$,
2. $d' \lhd\rhd_n e' \Rightarrow i_n^D(d') \lhd\rhd_{n+1} i_n^E(e')$,
3. $d \lhd\rhd_{n+1} a(S^\infty) \Rightarrow j_n^D(d) \lhd\rhd_n j_n^C(a)(S^\infty)$, *and*
4. $d' \lhd\rhd_n a'(S^\infty) \Rightarrow i_n^D(d') \lhd\rhd_{n+1} i_n^C(a')(S^\infty)$.

PROOF. By induction on n.

• $n = 0$: By definition, $j_0^D(d) = \perp_{D_0}$ for any $d \in D_1$, $j_0^E(e) = \perp_{E_1}$ for any $e \in E_1$, and $j_0^C(a)(S^\infty) = \perp_{E_0}$ for any $a \in A_1$. Therefore, $j_0^D(d) \lhd\rhd_0 j_0^E(e)$ and $j_0^D(d) \lhd\rhd_0 j_0^E(a)(S^\infty)$.

Likewise, $i_0^D(d') = \perp_{D_1}$ for any $d' \in D_0$, $i_0^E(e') = \perp_{E_1}$ for any $e' \in E_0$, and $i_0^C(a')(S^\infty) = \perp_{E_1}$ for any $a' \in A_0$. Therefore, $i_0^D(d') \lhd\rhd_1 i_0^E(e')$ and $i_0^D(d') \lhd\rhd_1 i_0^C(a')(S^\infty)$.

• $n > 0$:

(1). We assume $d \lhd\rhd_{n+1} e$, hence (by Lemma 6) either $d = \perp_{D_{n+1}}$ and $e = \perp_{E_{n+1}}$, or $d \neq \perp_{D_{n+1}}$ and $e \neq \perp_{E_{n+1}}$.

The first case is trivial by the definiton of j_n^D and j_n^E.

To prove the second case, let $d'' \in D_{n-1}$, and $a'' \in A_{n-1}$ such that $d'' \lhd\rhd_{n-1} a''(S^\infty)$. By induction hypothesis (4), we have that $i_{n-1}^D(d'') \lhd\rhd_n i_{n-1}^C(a'')(S^\infty)$. As $d \lhd\rhd_{n+1} e$, it must be that $d(i_{n-1}^D(d'')) \lhd\rhd_n e(i_{n-1}^C(a''))(S^\infty)$.
Then, by induction hypothesis (3),

$$j_{n-1}^D(d(i_{n-1}^D(d''))) \lhd\rhd_{n-1} j_{n-1}^C(e(i_{n-1}^C(a'')))(S^\infty),$$

and by definition of j_n^D and j_n^E we have that

$$j_n^D(d)(d'') \lhd\rhd_{n-1} j_n^E(e)(a'')(S^\infty).$$

Therefore we have proved that $j_n^D(d) \lhd\rhd_n j_n^E(e)$.

(4). We assume $d' \lhd\rhd_n a'(S^\infty)$, thus (by Lemma 6) either $d' = \perp_{D_n}$ and $a'(S^\infty) = \perp_{E_n}$, or $d' \neq \perp_{D_n}$ and $a'(S^\infty) \neq \perp_{E_n}$.

The first case is trivial by the definition of i_n^D and i_n^C.

For the second case, let $d'' \in D_n$ and $a'' \in A_n$ such that $d'' \lhd\rhd_n a''(S^\infty)$. By induction hypothesis (3),

$$j_{n-1}^D(d'') \lhd\rhd_{n-1} j_{n-1}^C(a'')(S^\infty).$$

As $d' \lhd\rhd_n a'(S^\infty)$, it must be that

$$d'(j_{n-1}^D(d'')) \lhd\rhd_{n-1} a'(S^\infty)(j_{n-1}^C(a''))(S^\infty).$$

Then, by induction hypothesis (4), we have that

$$i_{n-1}^D(d'(j_{n-1}^D(d''))) \lhd\rhd_n i_{n-1}^C(a'(S^\infty)(j_{n-1}^C(a')))(S^\infty),$$

and by definition of i_n^D and i_n^C we have that

$$i_n^D(d)(d'') \lhd\rhd_n i_n^C(a'(S^\infty))(a'')(S^\infty).$$

Therefore we have proved that $i_n^D(d') \lhd\rhd_{n+1} i_n^C(a')(S^\infty)$.
Proofs for (2) and (3) are similar to those for (1) and (4). □

The previous lemma enables to pass the similarity relation up and down through the approximations of D and E. We define a similarity relation between functions in D and E.

DEFINITION 2 (SIMILARITY). *$\lhd\rhd$ is defined as the least relation in $D \times E$ that verifies that for each $d \in D$ and $e \in E$, $d \lhd\rhd e$ if $\forall n \in \mathbb{N}$. $\psi_n^D(d) \lhd\rhd_n \psi_n^E(e)$.*

Similarity preserves undefinedness.

COROLLARY 8. *Let $d \in D$ and $e \in E$. If $d \lhd\rhd e$ then either $(d = \perp_D \wedge e = \perp_E)$ or $(d \neq \perp_D \wedge e \neq \perp_E)$.*

PROOF. Is a corollary of Lemma 6. □

We give an alternative characterization for $\lhd\rhd$ which is more convenient for writing proofs involving $\lhd\rhd$.

PROPOSITION 9. *Let $d \in D$, $e \in E$. $d \lhd\rhd e$ if and only if:*

- $(d = \perp_D \wedge e = \perp_E)$, *or*
- $(d \neq \perp_D \wedge e \neq \perp_E) \wedge \forall d' \in D. \forall a' \in [C \to E].$
  $d' \lhd\rhd a'(S^\infty) \Rightarrow d(d') \lhd\rhd e(a')(S^\infty)$.

PROOF. To simplify the notation we write r_n for $\psi_n^R(r)$ where $n \in \mathbb{N}$, $r \in R$ and $R \in \{D, E, [C \to E]\}$.
(if) We first prove that if d and e verify the property then they are similar.

- *Case 1:* $d = \perp_D$ and $e = \perp_E$.

We have that $(\perp_D)_n = \perp_{D_n}$ and $(\perp_E)_n = \perp_{E_n}$ for all $n \in \mathbb{N}$. By definiton of $\lhd\rhd_n$, $(\perp_D)_n = \perp_{D_n} \lhd\rhd_n \perp_{E_n} = (\perp_E)_n$, for any $n \in \mathbb{N}$. Therefore, $d \lhd\rhd e$.

- *Case 2:* $d \neq \perp_D$ and $e \neq \perp_E$.

By assumption, $d'' \lhd\rhd a''(S^\infty) \Rightarrow d(d'') \lhd\rhd e(a'')(S^\infty)$, for any $d'' \in D$ and any $a'' \in [C \to E]$.
By definition of $\lhd\rhd$, we can assure that if $d(d'') \lhd\rhd e(a'')(S^\infty)$, then $\forall m \in \mathbb{N}.(d(d''))_m \lhd\rhd_m (e(a'')(S^\infty))_m$. Then, by the equations (2) and (3) in Section 2.3, we have that
$\forall m \in \mathbb{N}. d_{m+1}(d''_m) \lhd\rhd_m e_{m+1}(a''_m)(S^\infty)$.
Consequently, for any $d'' \in D$ and any $a'' \in [C \to E]$,

$$d'' \lhd\rhd a''(S^\infty) \Rightarrow \forall m \in \mathbb{N}. d_{m+1}(d'_m) \lhd\rhd_m e_{m+1}(a'_m)(S^\infty). \quad (4)$$

Now we have to prove that $d \lhd\rhd e$, i.e., $\forall n \in \mathbb{N}. d_n \lhd\rhd_n e_n$.
But $d_0 = \perp_{D_0}$ and $e_0 = \perp_{E_0}$ and, by definition, $\perp_{D_0} \lhd\rhd_0 \perp_{E_0}$.
Thus, we have to check that $\forall n > 0. d_n \lhd\rhd_n e_n$, or equivalently, that $\forall n \in \mathbb{N}. d_{n+1} \lhd\rhd_{n+1} e_{n+1}$.
That is, it must be verified that for any $d' \in D_n$ and for any $a' \in A_n$ $d' \lhd\rhd_n a'(S^\infty) \Rightarrow d_{n+1}(d') \lhd\rhd_n e_{n+1}(a')(S^\infty)$.
Let $d' \in D_n$, we construct a tuple

$$\overline{d'} = \langle d'_0, d'_1, \ldots, d'_{n-1}, d', d'_{n+1} \ldots \rangle$$

such that $d'_k = j_k^D(d'_{k+1})$ for $k < n$ and $d'_k = i_{k-1}(d'_{k-1})$ for $k > n$. Therefore, $\overline{d'} \in D$.
Likewise, let $a' \in A_n = [C \to E_n]$, we construct

$$\overline{a'} = \langle a'_0, a'_1, \ldots, a'_{n-1}, a', a_{n+1} \ldots \rangle \in [C \to E].$$

By Lemma 7 we can assure that $\overline{d'} \lhd\rhd \overline{a'}(S^\infty)$ whenever $d' \lhd\rhd_n a'(S^\infty)$. Then, by (4) we have that

$$\forall m \in \mathbb{N}. d_{m+1}(\overline{d'}m) \lhd\rhd_m e_{m+1}(\overline{a'}m)(S^\infty),$$

and particularly: $d_{n+1}(d') \lhd\rhd_n e_{n+1}(a')(S^\infty)$.
(only if) Let us prove that if $d \lhd\rhd e$ then d and e satisfy the property. By Corollary 8 we only have two cases:

- *Case 1:* $d = \perp_D$ and $e = \perp_E$. Trivial.
- *Case 2:* $d \neq \perp_D$ and $e \neq \perp_E$.

Let $d' \in D$ and $a' \in [C \to E]$ such that $d' \lhd\rhd a'(S^\infty)$.
We have to prove that $d(d') \lhd\rhd e(a')(S^\infty)$, that is,
$\forall n \in \mathbb{N}.(d(d'))_n \lhd\rhd_n (e(a')(S^\infty))_n$.
For $n = 0$ this is trivial. Now consider $n > 0$; by hypothesis, $d \lhd\rhd e$ and $d' \lhd\rhd a'(S^\infty)$, therefore, by definition of $\lhd\rhd$ we have that $\forall m \in \mathbb{N}. d_m \lhd\rhd_m e_m \wedge d'_m \lhd\rhd_m a'_m(S^\infty)$.
Particularly: $d_{n+1} \lhd\rhd_{n+1} e_{n+1}$ and $d'_n \lhd\rhd_n a'_n(S^\infty)$.
Consequently, $d_{n+1}(d'_n) \lhd\rhd_n e_{n+1}(a'_n)(S^\infty)$, and therefore, $d(d')_n \lhd\rhd_n (e(a')(S^\infty))_n$, by (2) and (3) in Section 2.3. □

4. A DENOTATIONAL SEMANTICS FOR A LAZY λ-CALCULUS

The language described in [7] is a normalised λ-calculus extended with recursive *lets*. The restricted syntax is given in Figure 5, where all bound variables are distinct, and applications are of an expression to a variable.

We consider a *heap* Γ as a finite partial function from variables to expressions, that is, $Heap \in \mathcal{P}_f(Var \times Exp)$, with all the variables different pairwise.

4.1 A standard denotational semantics

To define a denotational semantics for this calculus, a domain of values and environments to associate values to the variables are needed.

An environment maps variables into values,

$$\rho \in Env = Var \to Value,$$

where *Value* is some appropriate domain containing at least a lifted version of its own function space, i.e.,

$$v \in Value = [Value \to Value]_\perp.$$

Notice that *Value* corresponds to the standard lifted domain D described in [2].

An ordering is defined on environments, such that if ρ is less or equal than ρ' then ρ' may bind more variables than ρ, but otherwise is equal to ρ. Formally, let $\rho, \rho' \in Env$ be environments, ρ is *less or equal* to ρ' (denoted as $\rho \leq \rho'$) iff $\forall x. \rho\ x \neq \perp \Rightarrow \rho\ x = \rho'\ x$.

For the language described by the syntax in Figure 5, Launchbury defines a denotational semantics in [7]. The semantic function is $[\![-]\!] : Exp \to Env \to Value$, which is given in Figure 6.

The function $\{\!\{-\}\!\} : Heap \to Env \to Env$ should be thought as an environment modifier defined as follows:

$$\{\!\{(x_i \mapsto e_i)_{i=1}^n\}\!\}\rho = \mu\rho'.\rho \sqcup (x_i \mapsto [\![e_i]\!]_{\rho'})_{i=1}^n,$$

where μ stands for the least fixed point operator and $\sqcup$ for

$$(\rho \sqcup (x \mapsto [\![e]\!]_{\rho'}))\ y = \begin{cases} \rho(y) & \text{if } y \neq x \\ [\![e]\!]_{\rho'} & \text{if } y = x \end{cases}$$

This definition only makes sense on environments ρ which are *consistent* with a heap Γ (i.e., if ρ and Γ bind the same variables, then they are bound to values for which an upper bound exists). This consistency is guaranteed in the denotational semantics definition in Figure 6 because we require that all bound variables are distinct.

$$
\begin{array}{rcll}
x & \in & \mathit{Var} & \\
e & \in & \mathit{Exp} & ::= \quad x \\
 & & & \mid \quad \lambda x.e \\
 & & & \mid \quad e\ x \\
 & & & \mid \quad \texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e
\end{array}
$$

Figure 5: Syntax

$$
\begin{array}{rl}
[\![x]\!]_\rho & = \rho\ x \\
[\![\lambda x.e]\!]_\rho & = \mathsf{up}(\lambda\nu.[\![e]\!]_{\rho \sqcup \{x \mapsto \nu\}}) \\
[\![e\ x]\!]_\rho & = \mathsf{AP}_D^\perp\ ([\![e]\!]_\rho, [\![x]\!]_\rho) \\
[\![\texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e]\!]_\rho & = [\![e]\!]_{\{\!(x_i \mapsto e_i)_{i=1}^n\}\!\}\rho}
\end{array}
$$

Figure 6: Denotational Semantics

4.2 A resourced denotational semantics

A resourced denotational semantics is also defined in [7], where the meaning of an expression depends on the number of available resources. For this a new domain of values, and the corresponding environments, are needed.

A *resourced environment* is a function mapping variables into functions from resources to values,

$$\sigma \in \mathit{CEnv} = \mathit{Var} \to [C \to \mathit{CValue}],$$

where *CValue* is some appropiate resourced domain, i.e.,

$$\mathit{CValue} = [[C \to \mathit{CValue}] \to [C \to \mathit{CValue}]]_\perp.$$

Notice that *CValue* corresponds to the resourced domain E constructed in Section 2.

An ordering is defined on resourced environments too, such that if σ is less or equal than σ' then, for any number of available resources, σ' may bind more variables than σ, but otherwise is equal to σ: Let $\sigma, \sigma' \in \mathit{CEnv}$ be resourced environments, σ is *less or equal* than σ' (denoted as $\sigma \leq \sigma'$) iff $\forall x \in \mathit{Var}$ and $\forall m \in \mathbb{N}$

$$\sigma x(S^m(\perp)) \neq \perp \Rightarrow \sigma\ x(S^m(\perp)) = \sigma' x(S^m(\perp)).$$

The resourced semantics focuses on approximations to the semantics of Figure 6. The semantic function $\mathcal{N}[\![-]\!] : \mathit{Exp} \to \mathit{CEnv} \to [C \to \mathit{CValue}]$ is given in Figure 7.

The function $\mathcal{N}\{\!\{-\}\!\} : \mathit{Heap} \to \mathit{CEnv} \to \mathit{CEnv}$ is defined analogously to $\{\!\{-\}\!\}$:

$$\mathcal{N}\{\!\{(x_i \mapsto E_i)_{i=1}^n\}\!\}\sigma = \mu\sigma'.\sigma \sqcup (x_i \mapsto \mathcal{N}[\![E_i]\!]_{\sigma'})_{i=1}^n.$$

4.3 Equivalence

In Section 3 we have shown how to relate values in *Value* with values in *CValue*, but we need to extend the notion of similarity to environments:

DEFINITION 3. (*Similarity of environments*). *Let $\rho \in \mathit{Env}$ be an environment and $\sigma \in \mathit{CEnv}$ be a resourced environment. ρ and σ are* similar *(denoted by $\rho \lhd\rhd \sigma$) when*

$$\forall x \in \mathit{Var}. \rho\ x \lhd\rhd \sigma\ x(S^\infty).$$

Now we can prove the equivalence between the standard denotational semantics and the resourced one. More precisely, we prove that they produce *similar* values.

THEOREM 10. *Let $e \in \mathit{Exp}$ be an expression and $\rho \in \mathit{Env}$, $\sigma \in \mathit{CEnv}$ be similar environments (i.e., $\rho \lhd\rhd \sigma$), then*

$$[\![e]\!]_\rho \lhd\rhd \mathcal{N}[\![e]\!]_\sigma\ (S^\infty).$$

PROOF. By structural induction on e:

$\mathbf{e \equiv x}$

By definition, $[\![x]\!]_\rho = \rho\ x$ and $\mathcal{N}[\![x]\!]_\sigma\ (S^\infty) = \sigma\ x(S^\infty)$.
By hypothesis $\rho \lhd\rhd \sigma$, therefore:
$[\![x]\!]_\rho = \rho\ x \lhd\rhd \sigma\ x(S^\infty) = \mathcal{N}[\![x]\!]_\sigma\ (S^\infty)$.

$\mathbf{e \equiv \lambda x.e'}$

By definition, $[\![\lambda x.e']\!]_\rho = \mathsf{up}(\lambda\nu.[\![e']\!]_{\rho \sqcup \{x \mapsto \nu\}}) \neq \perp_{\mathit{Value}}$ and
$\mathcal{N}[\![\lambda x.e']\!]_\sigma\ (S^\infty) = \mathsf{up}(\lambda\tau.\mathcal{N}[\![e']\!]_{\sigma \sqcup \{x \mapsto \tau\}}) \neq \perp_{\mathit{CValue}}$.
We have to prove that
$\mathsf{up}(\lambda\nu.[\![e']\!]_{\rho \sqcup \{x \mapsto \nu\}}) \lhd\rhd \mathsf{up}(\lambda\tau.\mathcal{N}[\![e']\!]_{\sigma \sqcup \{x \mapsto \tau\}})$.
We use the alternative characterization of similarity (Proposition 9). Let $v \in \mathit{Value}$ and $f \in C \to \mathit{CValue}$ such that $v \lhd\rhd f(S^\infty)$, then:
$\mathsf{AP}_D^\perp\ (\mathsf{up}(\lambda\nu.[\![e']\!]_{\rho \sqcup \{x \mapsto \nu\}}), v) = [\![e']\!]_{\rho \sqcup \{x \mapsto v\}}$ and
$\mathsf{AP}_E^\perp\ (\mathsf{up}(\lambda\tau.\mathcal{N}[\![e']\!]_{\sigma \sqcup \{x \mapsto \tau\}}), f, S^\infty) = \mathcal{N}[\![e']\!]_{\sigma \sqcup \{x \mapsto f\}}\ (S^\infty)$.
If $\rho' = \rho \sqcup \{x \mapsto v\} \lhd\rhd \sigma \sqcup \{x \mapsto f\} = \sigma'$, then, by induction hypothesis, we get the desired result.
Let us prove that $\rho' \lhd\rhd \sigma'$:

- If $y \neq x$ then $\rho'\ y = \rho\ y \lhd\rhd \sigma\ y(S^\infty) = \sigma'\ y$, because $\rho \lhd\rhd \sigma$.
- If $y = x$ then $\rho'\ y = v \lhd\rhd f(S^\infty) = \sigma'\ y$, by hypothesis.

Thus, $[\![e']\!]_{\rho'} \lhd\rhd \mathcal{N}[\![e']\!]_{\sigma'}\ (S^\infty)$.
Therefore, we have proved that $[\![\lambda x.e']\!]_\rho \lhd\rhd \mathcal{N}[\![\lambda x.e']\!]_\sigma\ (S^\infty)$.

$\mathbf{e \equiv e'\ x}$

By definition, $[\![e'\ x]\!]_\rho = \mathsf{AP}_D^\perp\ ([\![e']\!]_\rho, [\![x]\!]_\rho)$ and
$\mathcal{N}[\![e'\ x]\!]_\sigma\ (S^\infty) = \mathsf{AP}_E^\perp\ (\mathcal{N}[\![e']\!]_\sigma\ S^\infty, \mathcal{N}[\![x]\!]_\sigma, S^\infty)$.
Thus, we have to prove that
$\mathsf{AP}_D^\perp\ ([\![e']\!]_\rho, [\![x]\!]_\rho) \lhd\rhd \mathsf{AP}_E^\perp\ (\mathcal{N}[\![e']\!]_\sigma\ (S^\infty), \mathcal{N}[\![x]\!]_\sigma, S^\infty)$.
By induction hypothesis, $[\![e']\!]_\rho \lhd\rhd \mathcal{N}[\![e']\!]_\sigma\ (S^\infty)$. Then by Corollary 8 there are two cases:

- *Case 1:* $[\![e']\!]_\rho = \perp_{\mathit{Value}}$ and $\mathcal{N}[\![e']\!]_\sigma\ (S^\infty) = \perp_{\mathit{CValue}}$.
  $[\![e'\ x]\!]_\rho = \mathsf{AP}_D^\perp\ ([\![e']\!]_\rho, [\![x]\!]_\rho) = \mathsf{AP}_D^\perp\ (\perp_{\mathit{Value}}, [\![x]\!]_\rho) = \perp_{\mathit{Value}}$
  $\lhd\rhd \perp_{\mathit{CValue}} = \mathsf{AP}_E^\perp\ (\perp_{\mathit{CValue}}, \mathcal{N}[\![x]\!]_\sigma, S^\infty)$
  $= \mathsf{AP}_E^\perp\ (\mathcal{N}[\![e']\!]_\sigma\ (S^\infty), \mathcal{N}[\![x]\!]_\sigma, S^\infty) = \mathcal{N}[\![e'\ x]\!]_\sigma\ (S^\infty)$.
- *Case 2:* $\perp_{\mathit{Value}} \neq [\![e']\!]_\rho \lhd\rhd \mathcal{N}[\![e']\!]_\sigma\ (S^\infty) \neq \perp_{\mathit{CValue}}$.
  Thus, $\mathsf{AP}_D^\perp\ ([\![e']\!]_\rho, [\![x]\!]_\rho) = [\![e']\!]_\rho([\![x]\!]_\rho)$ and
  $\mathsf{AP}_E^\perp\ (\mathcal{N}[\![e']\!]_\sigma\ (S^\infty), \mathcal{N}[\![x]\!]_\sigma, S^\infty) =$
  $\mathcal{N}[\![e']\!]_\sigma\ (S^\infty)(\mathcal{N}[\![x]\!]_\sigma)\ (S^\infty)$.
  By induction hypothesis $[\![x]\!]_\rho \lhd\rhd \mathcal{N}[\![x]\!]_\sigma\ (S^\infty)$.
  Therefore, by the alternative characterization for $\lhd\rhd$:
  $[\![e']\!]_\rho([\![x]\!]_\rho) \lhd\rhd \mathcal{N}[\![e']\!]_\sigma\ (S^\infty)(\mathcal{N}[\![x]\!]_\sigma)\ (S^\infty)$.
  So that $[\![e'\ x]\!]_\rho \lhd\rhd \mathcal{N}[\![e'x]\!]_\sigma\ (S^\infty)$.

$\mathbf{e \equiv \texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e'}$

By definition:
$[\![\texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e']\!]_\rho = [\![e']\!]_{\{\!(x_i \mapsto e_i)_{i=1}^n\}\!\}\rho}$ and
$\mathcal{N}[\![\texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e']\!]_\sigma\ (S^\infty) =$
$\mathcal{N}[\![e']\!]_{\mathcal{N}\{\!(x_i \mapsto e_i)_{i=1}^n\}\!\}\sigma}\ (S^\infty)$.
Using fix point techniques it can be proved that if $\rho \lhd\rhd \sigma$ then $\{\!\{\Gamma\}\!\}\rho \lhd\rhd \mathcal{N}\{\!\{\Gamma\}\!\}\sigma$ for any heap Γ consistent with ρ and σ. Thus, $\{\!\{(x_i \mapsto e_i)_{i=1}^n\}\!\}\rho \lhd\rhd \mathcal{N}\{\!\{(x_i \mapsto e_i)_{i=1}^n\}\!\}\sigma$.
By induction hypothesis:
$[\![e']\!]_{\{\!(x_i \mapsto e_i)_{i=1}^n\}\!\}\rho} \lhd\rhd \mathcal{N}[\![e']\!]_{\mathcal{N}\{\!(x_i \mapsto e_i)_{i=1}^n\}\!\}\sigma}\ (S^\infty)$.
Therefore, we obtain that
$[\![\texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e']\!]_\rho \lhd\rhd$
$\mathcal{N}[\![\texttt{let}\ \{x_i = e_i\}_{i=1}^n\ \texttt{in}\ e']\!]_\sigma\ (S^\infty)$. □

This result allows to prove the computational adequacy of the natural semantics with respect to the denotational one.

$$
\begin{array}{rcl}
\mathcal{N}[\![e]\!]_\sigma\ (\bot) & = & \bot \\
\mathcal{N}[\![x]\!]_\sigma\ (S^{k+1}(\bot)) & = & \sigma\ x\ (S^k(\bot)) \\
\mathcal{N}[\![\lambda x.e]\!]_\sigma\ (S^{k+1}(\bot)) & = & \mathsf{up}(\lambda\tau.\mathcal{N}[\![e]\!]_{\sigma\sqcup\{x\mapsto\tau\}}\ ) \text{ where } \tau : C \to \mathit{CValue} \\
\mathcal{N}[\![e\ x]\!]_\sigma\ (S^{k+1}(\bot)) & = & \mathsf{AP}^{\bot}_{E}\ (\mathcal{N}[\![e]\!]_\sigma\ (S^k(\bot)), \mathcal{N}[\![x]\!]_\sigma\ , S^k(\bot)) \\
\mathcal{N}[\![\mathtt{let}\ \{x_i = e_i\}_{i=1}^n\ \mathtt{in}\ e]\!]_\sigma\ (S^{k+1}(\bot)) & = & \mathcal{N}[\![e]\!]_{\mathcal{N}\{\!\{(x_i\mapsto e_i)_{i=1}^n\}\!\}\sigma}\ (S^k(\bot))
\end{array}
$$

Figure 7: Resourced Denotational Semantics

5. CONCLUSIONS AND FUTURE WORK

We have tackled the problem of constructing a semantic domain for representing denotationally a resourced semantics. Abramsky and Ong defined in [2] an initial solution for the domain equation $D = [D \to D]_\perp$ suitable for defining a denotational semantics for a lazy λ-calculus. Following their schema, we have constructed the initial solution of the domain equation $E = [[C \to E] \to [C \to E]]_\perp$. We define a set of constructions on embeddings (adding a resource, lifting and function space), which can be cleanly defined by using function sections, and they simplify the presentation of the construction of E with respect to that of D in [2].

A resourced function space like E can be used to define a resourced denotational semantics that models the consumption of syntactic levels and thus, be nearer to syntax-oriented operational semantics based on rules, such as Launchbury's natural semantics for lazy evaluation. In fact, this is the approach taken by Launchbury in [7] in order to prove the computational adequacy of his natural semantics. But the equivalence between the standard and the resourced denotational semantics turns out to be not so easy to establish, because the semantic domains are different. The same problem can be found in [15].

To prove the computational adequacy it is only required that both semantics converge for the same expressions. However, in order to prove this result by structural induction a stronger property is needed, namely that both semantics produce values that behave "similarly". The problem arises in the application rule, because it depends on the semantics of the functional abstraction. Therefore, we have defined a similarity relation between the values of $D = [D \to D]_\perp$ and those of $E = [[C \to E] \to [C \to E]]_\perp$. Since a direct definition of the relation between D and E is not possible, we have first defined the similarity gradually between the approximation domains D_n and E_n. Afterwards, we prove that this layered definition satisfies an applicative bisimulation-like property.

We are interested in studying more properties of the similarity relation, especially in the context of category theory. We also want to compare it with the notions of bisimulation and bisimilarity —particularly with Abramsky's applicative bisimulation [1]— and with observational or contextual equivalences, like those in [10] or [12].

6. ACKNOWLEDGMENTS

We are thankful to John Launchbury for his valuable comments, and appreciate very much the help of David de Frutos Escrig. This work is partially supported by the grants: TIN2009-14599-C03-01, BES-2007-16823, S2009/TIC-1465.

7. REFERENCES

[1] S. Abramsky. *Research Topics in Functional Programming*, chapter The Lazy Lambda Calculus, pages 65–116. Addison-Wesley, 1990.

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

[3] C. Baker-Finch, D. King, and P. W. Trinder. An operational semantics for parallel lazy evaluation. In *ACM International Conference on Functional Programming (ICFP'00)*, pages 162–173, 2000.

[4] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[5] G. Boudol, P. Curien, and C. Lavatelli. A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9(4):437–482, 1999.

[6] C. Gunter and D. S. Scott. Semantic domains. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 633–674. Elsevier Science, 1990.

[7] J. Launchbury. A natural semantics for lazy evaluation. In *ACM Symposium on Principles of Programming Languages (POPL'93)*, pages 144–154. ACM Press, 1993.

[8] A. Melton, D. A. Schmidt, and G. E. Strecker. Galois connections and computer science applications. In *Category Theory and Computer Programming*, pages 299–312. LNCS 240, Springer, 1986.

[9] K. Nakata and M. Hasegawa. Small-step and big-step semantics for call-by-need. *Journal of Functional Programming*, 19(6):699–722, 2009.

[10] G. D. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Journal of Theoretical Computer Science*, 1(2):125–159, 1975.

[11] L. Sánchez-Gil, M. Hidalgo-Herrero, and Y. Ortega-Mallén. *Trends in Functional Programming*, volume 10, chapter An Operational Semantics for Distributed Lazy Evaluation, pages 65–80. Intellect, 2010.

[12] M. Schmidt-Schauğ. Equivalence of call-by-name and call-by-need for lambda-calculi with letrec. Technical report, Institut für Informatik. J. W. Goethe Universität Frankfurt am Main, Germany, 2006.

[13] D. S. Scott. Continuous lattices. In *Toposes, Algebraic Geometry and Logic*, pages 97–136. LNCS 274, Springer, 1972.

[14] P. Sestoft. Deriving a lazy abstract machine. *Journal of Functional Programming*, 7(3):231–264, 1997.

[15] M. van Eekelen and M. de Mol. *Reflections on Type Theory, λ-calculus, and the Mind. Essays dedicated to Henk Barendregt on the Occasion of his 60th Birthday*, chapter Proving Lazy Folklore with Mixed Lazy/Strict Semantics, pages 87–101. Radboud University Nijmegen, The Netherlands, 2007.