Math. Struct. in Comp. Science: *page* 1 of 35. © Cambridge University Press 2017
doi:10.1017/S0960129516000396

Execution time of λ-terms via denotational semantics and intersection types

DANIEL DE CARVALHO

Innopolis University, Universitetskaya St, 1, Innopolis, 420500, Tatarstan, Russia
Email: d.carvalho@innopolis.ru

Received 28 February 2010; revised 21 January 2011

The multiset-based relational model of linear logic induces a semantics of the untyped λ-calculus, which corresponds with a non-idempotent intersection type system, System R. We prove that, in System R, the size of type derivations and the size of types are closely related to the execution time of λ-terms in a particular environment machine, Krivine's machine.

1. Introduction

This paper presents a way to extract information on execution time of λ-terms by semantic means. This work is inspired by Ehrhard and Regnier (2006b), which considers the resource λ-calculus, a resource-oriented version of λ-calculus introduced in Ehrhard and Regnier (2006a), similar to the versions introduced in Boudol *et al.* (1999) and Kfoury (2000). In this paper, the resource λ-calculus does not appear explicitly. Nevertheless, we evoke in Subsection 4.3 a precise relation between resource λ-terms and the objects we consider.

By execution time, we mean the number of steps in a computational model. As in Ehrhard and Regnier (2006b), the computational model considered in this paper will be Krivine's machine (Krivine 2007), a more realistic model than β-reduction. Indeed, Krivine's machine implements (weak) head linear reduction: In one step, we can do at most one substitution. In this paper, we consider two variants of this machine: The original machine is extended to carry on the execution under abstractions, the first one (Definition 2.4) computes the head-normal form of any λ-term (if it exists) and the second one (Definition 2.11) computes the normal form of any λ-term (if it exists). Section 2 presents these several notions of call-by-name execution of λ-terms.

The fundamental idea of denotational semantics is that types should be interpreted as objects of a category $\mathbb{C}$ and terms should be interpreted as arrows in $\mathbb{C}$ in such a way that if a term t reduces to a term t', then they are interpreted by the same arrow. By the Curry–Howard isomorphism, a simply typed λ-term is a proof in intuitionistic logic and the β-reduction of a λ-term corresponds to the cut-elimination of a proof. Now, the intuitionistic fragment of linear logic (Girard 1987) is a refinement of intuitionistic logic. This means that when we have a categorical structure $(\mathbb{C}, \ldots)$ for interpreting intuitionistic linear logic, we can derive a category $\mathbb{K}$ that is a denotational semantics of intuitionistic logic and thus a denotational semantics of λ-calculus.

Linear logic has various denotational semantics; one amongst these is the multiset-based relational model in the category **Rel** of sets and relations with the comonad associated with the finite multisets functor (see Tortora de Falco (2000) for interpretations of proof-nets and Appendix of Bucciarelli and Ehrhard (2001) for interpretations of derivations of sequent calculus). In this paper, the category $\mathbb{K}$ is a category equivalent to the co-Kleisli category of this comonad. The semantics we obtain is *non-uniform* in the following sense: The interpretation of a function contains information about its behaviour on 'chimerical' (i.e. non-deterministic) arguments (see Example 3.8 for an illustration of this fact). As we want to consider type-free λ-calculus, we will consider a λ-algebra in $\mathbb{K}$. Section 3 presents this λ-algebra, which was obtained as a reflexive object in the cartesian closed category $\mathbb{K}$ derived from the multiset-based relational model of linear logic.

In Section 4, we describe the semantics of λ-terms in this λ-algebra as a logical system, using intersection types. The intersection types system that we consider (System R, defined in Subsection 4.1) is a reformulation of that of Coppo et al. (1980); in particular, it lacks idempotency, as System λ in Kfoury (2000) and System $\mathbb{I}$ in Neergaard and Mairson (2004) and contrary to System $\mathcal{I}$ of Kfoury et al. (1999). Basically, types are points in the λ-algebra and the typing rules reflect the interpretation of λ-terms; it follows that the semantics of a closed λ-term is just the set of its types in System R. As such, we stress the fact that the semantics of Coppo et al. (1980) can be reconstructed in a natural way from the finite multisets relational model of linear logic using the co-Kleisli construction.

If t' is a λ-term obtained by applying some reduction steps to t, then the semantics $[\![t']\!]$ of t' and the semantics $[\![t]\!]$ of t are equal, so that from $[\![t]\!]$, it is clearly impossible to determine the number of reduction steps leading from t to t'. Nevertheless, if v and u are two closed normal λ-terms, we can pose the following questions:

1. Is it the case that the λ-term $(v)u$ is head-normalisable? Normalisable?
2. If the answer to the previous question is positive, what is the number of steps leading to the (principal head) normal form?

The main points of the current paper are to show that:

— it is possible to answer both questions by only referring to the semantics $[\![v]\!]$ and $[\![u]\!]$ of v and u, respectively;

— and one can define in the semantics a natural measure, which is induced by the cardinalities of the multisets, that corresponds to this number of steps.

The answer to the first question is given in Section 5 (Corollary 5.7), which provides characterisations of normalisability (Proposition 5.6):

— A λ-term is head-normalisable if and only if it is typable in System R.

— A λ-term is normalisable if and only if it admits in System R a type with no positive occurrence of the empty multiset.

Such properties are a simple adaptation of well-known results.

The answer to the second question is given in Section 6. The paper (Ronchi Della Rocca 1988) presents a procedure that computes a normal form of any λ-term (if it exists) by finding its principal typing (if it exists). In Section 6, we present some quantitative results about the relation between the types and the computation of the (principal head) normal form. First, we extend System R to the states of the machines introduced in Section 2. Then, we prove that the number of steps of execution of a λ-term in the first machine (from Definition 2.4) is the size of the least type derivation of the λ-term in System R (Theorem 6.9) and we prove a similar result (Theorem 6.15) for the second machine (from Definition 2.11). We end by proving truly semantic measures of execution time in Subsection 6.4 and Subsection 6.5.

Note that even if this paper, a revised version of de Carvalho (2006), concerns the λ-calculus and Krivine's machine, we emphasize connections with proof-nets of linear logic. Due to these connections, we conjectured in de Carvalho (2007) that we could obtain some similar results relating on the one hand the length of cut-elimination of nets with some specific strategy and on the other hand the size of the results of experiments (experiments, introduced in Girard (1987), are functions defined on proof-nets allowing to compute the interpretation pointwise, the set of *results* of all the experiments of a given proof-net is its interpretation). This specific strategy should be a strategy that mimics that of Krivine's machine and that extends the strategy defined in Mascari and Pedicini (1994) for a fragment of linear logic. This work has been done in de Carvalho et al. (2008) by adapting our work for the λ-calculus. Nonetheless, it is difficult to compare both works because in the syntax of proof-nets we considered, a cut-elimination step is not as elementary as a reduction step in Krivine's machine.

In conclusion, we believe that this work can be useful for implicit characterisations of complexity classes (in particular, the PTIME class, as in Baillot and Terui (2004)) by providing a semantic setting in which quantitative aspects can be studied, whilst taking some distance from the syntactic details.

In summary, Section 2 presents Krivine's machine, Section 3 the semantics we consider, and Section 4 the intersection type system induced by this semantics, namely System R; Section 5 gives the answer to question (1) and Section 6 the answer to question (2).

Notations. We denote by Λ the set of λ-terms, by $\mathcal{V}$ the set of variables and, for any λ-term t, by $FV(t)$ the set of free variables in t.

We use Krivine's notation for λ-terms: The λ-term v applied to the λ-term u is denoted by $(v)u$. We will also denote $(\dots((v)u_1)u_2\dots)u_k$ by $(v)u_1\dots u_k$.

We use the notation $[]$ for multisets. For any set A, we denote by $\mathcal{M}(A)$ the set of finite multisets a whose support, denoted by $\mathsf{Supp}(a)$, is a subset of A. For any set A, for any $n \in \mathbb{N}$, we denote by $\mathcal{M}_n(A)$ the set of multisets of cardinality n whose support is a subset of A. The binary union of multisets given by term-by-term addition of multiplicities is denoted by a $+$ sign and, following this notation, the generalised union is denoted by a $\sum$ sign. The neutral element for this operation, the empty multiset, is denoted by $[]$.

For any set A and for any $a \in A$, we will use the following convention: If $m = 0$, then $(\prod_{j=1}^{m} A_j) \times A = A$ and $((a_1, \dots, a_m), a) = a$.

For any set A, for any $n \in \mathbb{N}$, for any $a \in \mathcal{M}_n(A)$, we set

$$\mathfrak{S}(a) = \{(\alpha_1, \dots, \alpha_n) \in A^n \ / \ a = [\alpha_1, \dots, \alpha_n]\};$$

for instance, $\mathfrak{S}([\alpha, \alpha, \beta]) = \{(\alpha, \alpha, \beta), (\alpha, \beta, \alpha), (\beta, \alpha, \alpha)\}$.

For any sets A and B, we denote by $A \rightharpoonup_{\text{fin}} B$ the set of partial functions from A to B whose domain is finite.

2. Krivine's machine

We introduce two variants of a machine presented in Krivine (2007) that implements call-by-name. More precisely, the original machine performs weak head linear reduction (this notion is defined in Danos and Regnier (2004)), whereas the machine presented in Subsection 2.2 performs head linear reduction. Subsection 2.3 slightly modifies the latter machine as to compute the β-normal form of any normalisable term.

2.1. *Execution of states*

We begin with the definition of the set $\mathcal{E}$ of environments. We define, by induction on p, a set $\mathcal{E}_p$ for any $p \in \mathbb{N}$:

— if $p = 0$, then $\mathcal{E}_p = \mathcal{V} \rightharpoonup_{\text{fin}} \emptyset$;

— for any $p \in \mathbb{N}$, $\mathcal{E}_{p+1} = \mathcal{V} \rightharpoonup_{\text{fin}} \Lambda \times \mathcal{E}_p$.

We define, by induction on p, a function $j_{p,q} : \mathcal{E}_p \to \mathcal{E}_q$ for any $p, q \in \mathbb{N}$ such that $p \leqslant q$:

— if $p = 0$, then $j_{p,q}(e)$ is the partial function in $\mathcal{E}_q$ whose domain is empty;

— for any $p, q \in \mathbb{N}$ such that $p \leqslant q$, we set $j_{p+1,q+1}(e) = (id_\Lambda \times j_{p,q}) \circ e$.

Let $(\mathcal{E}_p \xrightarrow{i_p} \mathcal{E})_{p \in \mathbb{N}}$ be a direct limit of the direct system $((\mathcal{E}_p)_{p \in \mathbb{N}}, (j_{p,q})_{p,q \in \mathbb{N}, p \leqslant q})$ over $(\mathbb{N}, \leqslant)$ in the category of sets. For instance, if the partial functions are identified with their graphs, then the $j_{p,q}$ are the inclusions from $\mathcal{E}_p$ to $\mathcal{E}_q$, we have $\mathcal{E} = \bigcup_{p \in \mathbb{N}} \mathcal{E}_p$ and the i_p are the inclusions from $\mathcal{E}_p$ to $\mathcal{E}$.

We have $\mathcal{E} = \bigcup_{p \in \mathbb{N}} i_p(\mathcal{E}_p)$, hence, for any $e \in \mathcal{E}$, we can denote by $\mathsf{d}(e)$ the least integer p such that $e \in i_p(\mathcal{E}_p)$; moreover, each i_p is injective, so there exists a unique $e_0 \in \mathcal{E}_{\mathsf{d}(e)}$ such that $i_{\mathsf{d}(e)}(e_0) = e$: We denote by $\mathsf{dom}(e)$ the domain of e_0 and by $\mathsf{im}(e)$ the image of e_0, and we set $e(x) = e_0(x)$ for any $x \in \mathsf{dom}(e)$. We denote by $\perp$ the unique environment e such that $\mathsf{dom}(e) = \emptyset$. We denote by $\{x_1 \mapsto c_1, \ldots, x_m \mapsto c_m\}$ the unique environment e such that $\mathsf{dom}(e) = \{x_1, \ldots, x_m\}$ and $e(x_1) = c_1, \ldots, e(x_m) = c_m$. If e_1 and e_2 are two environments such that $\mathsf{dom}(e_1) \cap \mathsf{dom}(e_2) = \emptyset$, we denote by $e_1 \cup e_2$ the environment e such that $\mathsf{dom}(e) = \mathsf{dom}(e_1) \cup \mathsf{dom}(e_2)$ and, for any $x \in \mathsf{dom}(e)$, we have
$$e(x) = \begin{cases} e_1(x) & \text{if } x \in \mathsf{dom}(e_1); \\ e_2(x) & \text{if } x \in \mathsf{dom}(e_2). \end{cases}$$

The set $\mathcal{C}$ of closures is defined as follows: $\mathcal{C} = \Lambda \times \mathcal{E}$.

For $c = (t, e) \in \mathcal{C}$, we define $\overline{c} = t[e] \in \Lambda$ by induction on $\mathsf{d}(e)$:

— If $\mathsf{d}(e) = 0$, then $t[e] = t$.

— Assume $t[e]$ defined for $\mathsf{d}(e) = d$. If $\mathsf{d}(e) = d + 1$, then $t[e] = t[\overline{e(x_1)}/x_1, \ldots, \overline{e(x_m)}/x_m]$, where $\{x_1, \ldots, x_m\} = \mathsf{dom}(e)$.

A *stack* is a finite sequence of closures. If c_0 is a closure and $\pi = (c_1, \ldots, c_q)$ is a stack, then $c_0 \cdot \pi$ will denote the stack $(c_0, \ldots, c_q)$. We will denote by ε the empty stack.

A *state* is a non-empty stack. If $s = (c_0, \ldots, c_q)$ is a state, then $\overline{s}$ will denote the λ-term $(\overline{c_0})\overline{c_1} \ldots \overline{c_q}$.

Definition 2.1. For any closure (t, e), we define, by induction on $\mathsf{d}(e)$, what means that *the closure* (t, e) *respects the variable convention*: The closure (t, e) respects the variable convention if and only if

— any bound variable in t is bound in t at most once;
— for any bound variable x in t, we have $x \notin \mathsf{dom}(e)$;
— and, for any $c \in \mathsf{im}(e)$, the closure c respects the variable convention.

We say that a state $(c_0, \ldots, c_q)$ respects the variable convention if and only if the closures $c_0, \ldots, c_q$ respect the variable convention.

We denote by $\mathbb{S}$ the set of the states that respect the variable convention.

First, we present the execution of a state (that respects the variable convention). It consists in updating a closure (t, e) and a stack. If t is an application $(v)u$, then we push the closure (u, e) on the top of the stack and the current closure is now (v, e). If t is an abstraction, then a closure is popped and a new environment is created. If t is a variable, then the current closure is now the value of the environment at the variable. The partial map $s \succ_{\mathbb{S}} s'$ (defined below) defines formally the transition from a state to another state.

Definition 2.2. We define a partial map from $\mathbb{S}$ to $\mathbb{S}$: For any $s, s' \in \mathbb{S}$, the notation $s \succ_{\mathbb{S}} s'$ will mean that the map assigns s' to s. The value of the map at s is defined as follows:

$$s \mapsto \begin{cases} e(x) \cdot \pi & \text{if } s = (x, e) \cdot \pi \text{ and } x \in \mathsf{dom}(e); \\ (u, e \cup \{x \mapsto c\}) \cdot \pi' & \text{if } s = (\lambda x.u, e) \cdot (c \cdot \pi'); \\ (v, e) \cdot ((u, e) \cdot \pi) & \text{if } s = ((v)u, e) \cdot \pi. \end{cases}$$

Note that in the case where the current subterm is an abstraction and the stack is empty, the machine stops: It does not reduce under lambda abstractions. That is why we slightly modify this machine in the following subsection.

2.2. *A machine computing the principal head normal form*

Now, the machine has to reduce under lambda abstractions and, in Subsection 2.3, the machine will have to compute the arguments of the head variable. So, we extend the machine so that it performs the reduction of elements of $\mathcal{K}$, where $\mathcal{K} = \bigcup_{n \in \mathbb{N}} \mathcal{K}_n$ with

— $\mathcal{H}_0 = \mathcal{V}$ and $\mathcal{K}_0 = \mathbb{S}$;
— $\mathcal{H}_{n+1} = \mathcal{V} \cup \{(v)u \,/\, v \in \mathcal{H}_n \text{ and } u \in \Lambda \cup \mathcal{K}_n\}$ and
  $\mathcal{K}_{n+1} = \mathbb{S} \cup \mathcal{H}_n \cup \{\lambda y.k \,/\, y \in \mathcal{V} \text{ and } k \in \mathcal{K}_n\}$.

Set $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$. We have $\mathcal{K} = \mathbb{S} \cup \mathcal{H} \cup \{\lambda x.k \,/\, x \in \mathcal{V} \text{ and } k \in \mathcal{K}\}$.

Remark 2.3. We have

— $\mathcal{H} = \bigcup_{p \in \mathbb{N}} \{(x)t_1 \ldots t_p \,/\, x \in \mathcal{V} \text{ and } t_1, \ldots, t_p \in \Lambda \cup \mathcal{K}\}$
— hence, any element of $\mathcal{K}$ can be written as either

$$\lambda x_1 \ldots \lambda x_m.s \text{ with } m \in \mathbb{N},\ x_1, \ldots, x_m \in \mathcal{V} \text{ and } s \in \mathbb{S}$$

or

$$\lambda x_1 \ldots \lambda x_m.(x)t_1 \ldots t_p \text{ with } m, p \in \mathbb{N},\ x_1, \ldots, x_m \in \mathcal{V} \text{ and } t_1, \ldots, t_p \in \mathcal{K} \cup \Lambda.$$

For any $k \in \mathcal{K}$, we denote by $\mathsf{d}(k)$ the least integer p such that $k \in \mathcal{K}_p$.

We extend the definition of $\overline{s}$ for $s \in \mathbb{S}$ to $\overline{k}$ for $k \in \mathcal{K}$. For that, we set $\overline{t} = t$ if $t \in \Lambda$. This definition is by induction on $\mathsf{d}(k)$:

— if $\mathsf{d}(k) = 0$, then $k \in \mathbb{S}$ and thus $\overline{k}$ is already defined;

— if $k \in \mathcal{H}$, then there are two cases:

 – if $k \in \mathcal{V}$, then $\overline{k}$ is already defined (it is k);

 – else, $k = (v)u$ and we set $\overline{k} = (\overline{v})\overline{u}$;

— if $k = \lambda x.k_0$, then $\overline{k} = \lambda x.\overline{k_0}$.

Definition 2.4. We define a partial map from $\mathcal{K}$ to $\mathcal{K}$: For any $k, k' \in \mathcal{K}$, the notation $k \succ_h k'$ will mean that the map assigns k' to k. The value of the map at k is defined, by induction on $\mathsf{d}(k)$, as follows:

$$k \mapsto \begin{cases} s' & \text{if } k \in \mathbb{S} \text{ and } k \succ_{\mathbb{S}} s'; \\ (x)\overline{c_1}\ldots\overline{c_q} & \text{if } k = ((x,e), c_1, \ldots, c_q) \in \mathbb{S} \text{ and } x \in \mathcal{V} \setminus \mathsf{dom}(e); \\ \lambda x.((u,e)\cdot\varepsilon) & \text{if } k = (\lambda x.u, e)\cdot\varepsilon \in \mathbb{S}; \\ \lambda y.k'_0 & \text{if } k = \lambda y.k_0 \text{ and } k_0 \succ_h k'_0. \end{cases}$$

A difference with the original machine is that this machine reduces under lambda abstractions. Moreover, when the state is a variable not declared in the environment, it does one (non-elementary) step more, but this other difference is just a device in order to have an output in a nicer form.

We denote by $\succ_h{}^*$ the reflexive transitive closure of $\succ_h$. For any $k \in \mathcal{K}$, k is said to be a *Krivine head normal form* if for any $k' \in \mathcal{K}$, we do not have $k \succ_h k'$.

Definition 2.5. For any $k_0 \in \mathcal{K}$, we define $l_h(k_0) \in \mathbb{N} \cup \{\infty\}$ as follows: If there exist $k_1, \ldots, k_n \in \mathcal{K}$ such that $k_i \succ_h k_{i+1}$ for $0 \leqslant i \leqslant n-1$ and k_n is a Krivine head normal form, then we set $l_h(k_0) = n$, else we set $l_h(k_0) = \infty$.

Proposition 2.6. For any $s \in \mathbb{S}$, for any Krivine head normal form $k' \in \mathcal{K}$ such that $s \succ_h{}^* k'$, k' is a λ-term in head normal form.

Proof. By induction on $l_h(s)$. The base case is trivial, because we never have $l_h(s) = 0$.

The inductive step is divided into five cases.

— If $s = (x,e)\cdot(c_1,\ldots,c_q)$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, then $s \succ_h (x)\overline{c_1}\ldots\overline{c_q}$. But $(x)\overline{c_1}\ldots\overline{c_q}$ is a Krivine head normal form and a λ-term in head normal form.

— If $s = (x,e)\cdot\pi$ and $x \in \mathsf{dom}(e)$, then $s \succ_h e(x)\cdot\pi$. Now, $e(x)\cdot\pi \in \mathbb{S}$ and $e(x)\cdot\pi \succ_h{}^* k'$, hence, by induction hypothesis, k' is a λ-term in head normal form.

— If $s = (\lambda x.u, e)\cdot\varepsilon$, then $k' = \lambda x.k''$ with $(u,e)\cdot\varepsilon \succ_h{}^* k''$. Now, $(u,e)\cdot\varepsilon \in \mathbb{S}$, hence, by induction hypothesis, k'' is a λ-term in head normal form; therefore, k' too is a λ-term in head normal form.

— If $s = (\lambda x.u, e)\cdot(c\cdot\pi)$, then $s \succ_h (u, e \cup \{x \mapsto c\})\cdot\pi$. Now, $(u, e \cup \{x \mapsto c\})\cdot\pi \in \mathbb{S}$ and $(u, e \cup \{x \mapsto c\})\cdot\pi \succ_h{}^* k'$, hence, by induction hypothesis, k' is a λ-term in head normal form.

	output	current subterm	environment	stack
		$(\lambda x.(x)x)\lambda y.y$	$\bot$	ε
1		$\lambda x.(x)x$	$\bot$	$(\lambda y.y,\bot)$
2		$(x)x$	$\{x\mapsto(\lambda y.y,\bot)\}$	ε
3		x	$\{x\mapsto(\lambda y.y,\bot)\}$	$(x,\{x\mapsto(\lambda y.y,\bot)\})$
4		$\lambda y.y$	$\bot$	$(x,\{x\mapsto(\lambda y.y,\bot)\})$
5		y	$\{y\mapsto(x,\{x\mapsto(\lambda y.y,\bot)\})\}$	ε
6		x	$\{x\mapsto(\lambda y.y,\bot)\}$	ε
7		$\lambda y.y$	$\bot$	ε
8	$\lambda y.$	y	$\bot$	ε
9	$\lambda y.y$			

Fig. 1. Example of computation of the principal head normal form.

— If $s=((v)u,e)\cdot\pi$, then $s\succ_h(v,e)\cdot((u,e)\cdot\pi)$. Now, $(v,e)\cdot((u,e)\cdot\pi)\in\mathbb{S}$ and $(v,e)\cdot((u,e)\cdot\pi){\succ_h}^*k'$, hence, by induction hypothesis, k' is a λ-term in head normal form. □

Example 2.7. Set $s=(((\lambda x.(x)x)\lambda y.y,\bot),\varepsilon)$. We have $l_h(s)=9$:

$$\begin{aligned} s &\succ_h ((\lambda x.(x)x,\bot),(\lambda y.y,\bot))\\ &\succ_h ((x)x,\{x\mapsto(\lambda y.y,\bot)\}).\varepsilon\\ &\succ_h ((x,\{x\mapsto(\lambda y.y,\bot)\}),(x,\{x\mapsto(\lambda y.y,\bot)\}))\\ &\succ_h ((\lambda y.y,\bot),(x,\{x\mapsto(\lambda y.y,\bot)\}))\\ &\succ_h (y,\{y\mapsto(x,\{x\mapsto(\lambda y.y,\bot)\})\})\cdot\varepsilon\\ &\succ_h (x,\{x\mapsto(\lambda y.y,\bot)\})\cdot\varepsilon\\ &\succ_h (\lambda y.y,\bot)\cdot\varepsilon\\ &\succ_h \lambda y.((y,\bot)\cdot\varepsilon)\\ &\succ_h \lambda y.y\end{aligned}$$

We present the same computation in a more descriptive way in Figure 1.

Lemma 2.8. For any $k,k'\in\mathcal{K}$, if $k\succ_h k'$, then $\overline{k}\rightarrow_h\overline{k'}$, where $\rightarrow_h$ is the reflexive closure of the head reduction.

Proof. By induction on $\mathsf{d}(k)$. There are two cases:

— If $k\in\mathbb{S}$, then there are five cases:

– If $k=(x,e)\cdot(c_1,\dots,c_q)$ and $x\in\mathsf{dom}(e)$, then $\overline{k}=\overline{e(x)}\overline{c_1}\dots\overline{c_q}$ and $\overline{k'}=\overline{e(x)\cdot(c_1,\dots,c_q)}=(\overline{e(x)})\overline{c_1}\dots\overline{c_q}$: we have $\overline{k}=\overline{k'}$.

– If $k=(\lambda x.u,e)\cdot(c_0,\dots,c_q)$, then $\overline{k}=((\lambda x.u)[e])\overline{c_0}\dots\overline{c_q}=(\lambda x.u[e])\overline{c_0}\dots\overline{c_q}$ (since k respects the variable convention) and $\overline{k'}=(\overline{(u,e\cup\{x\mapsto c_0\})})\overline{c_1}\dots\overline{c_q}$. Now, $\overline{k}$ reduces in a single head reduction step to $(u[e][\overline{c_0}/x])\overline{c_1}\dots\overline{c_q}=\overline{k'}$.

- If $k = ((v)u, e) \cdot (c_1 \dots c_q)$, then $\overline{k} = (((v)u)[e])\overline{c_1} \dots \overline{c_q} = (v[e])u[e]\overline{c_1} \dots \overline{c_q}$ and $\overline{k'} = \overline{(v, e) \cdot ((u, e), c_1, \dots, c_q)} = (v[e])u[e]\overline{c_1} \dots \overline{c_q}$: we have $\overline{k} = \overline{k'}$.
- If $k = (x, e) \cdot (c_1, \dots, c_q)$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, then $\overline{k} = (x)\overline{c_1} \dots \overline{c_q}$ and $\overline{k'} = \overline{(x)c_1 \dots c_q} = (x)\overline{c_1} \dots \overline{c_q}$, we have $\overline{k} = \overline{k'}$.
- If $k = (\lambda x.u, e).\varepsilon$, then $\overline{k} = (\lambda x.u)[e] = \lambda x.u[e]$ (because k respects the variable convention) and $\overline{k'} = \overline{\lambda x.((u, e), \varepsilon)} = \lambda x.u[e]$, we have $\overline{k} = \overline{k'}$.

— Otherwise, $k = \lambda y.k_0$; then $\overline{k} = \lambda y.\overline{k_0}$ and $\overline{k'} = \overline{\lambda y.k'_0} = \lambda y.\overline{k'_0}$ with $k_0 \succ_h k'_0$: by induction hypothesis, we have $\overline{k_0} \rightarrow_h \overline{k'_0}$, hence $\overline{k} \rightarrow_h \overline{k'}$. □

Theorem 2.9. For any $k \in \mathcal{K}$, if $l_h(k)$ is finite, then $\overline{k}$ is head normalisable.

Proof. By induction on $l_h(k)$.
If $l_h(k) = 0$, then $\overline{k}$ is a head normal form. Otherwise, apply Lemma 2.8. □

For any head normalisable λ-term t, we denote by $\mathsf{h}(t)$ the number of head reductions of t.

Theorem 2.10. For any $s = (t, e) \cdot \pi \in \mathbb{S}$, if $\overline{s}$ is head normalisable, then $l_h(s)$ is finite.

Proof. We prove, by Noetherian induction on $(\mathsf{h}(\overline{s}), d(e), t)$ with respect to the lexicographical order on $\mathbb{N} \times \mathbb{N} \times \Lambda$, that for any $s = (t, e) \cdot \pi$ such that $\overline{s}$ is head-normalisable, $l_h(s)$ is finite.
If $\mathsf{h}(\overline{s}) = 0$, $\mathsf{d}(e) = 0$ and $t \in \mathcal{V}$, then we have $l_h(s) = 1$.
Otherwise, there are five cases.

— In the case where $t \in \mathsf{dom}(e)$, we have $s \succ_h e(t) \cdot \pi$. Set $s' = e(t) \cdot \pi$ and $e(t) = (t', e')$. We have $\overline{s} = \overline{s'}$ and $\mathsf{d}(e') < \mathsf{d}(e)$, hence we can apply the induction hypothesis: $l_h(s')$ is finite and thus $l_h(s) = l_h(s') + 1$ is finite.
— In the case where $t = \lambda x.u$ and $\pi = c \cdot \pi'$, we have $s \succ_h (u, e \cup \{x \mapsto c\}) \cdot \pi'$. Set $s' = (u, e \cup \{x \mapsto c\}) \cdot \pi'$. We have $\mathsf{h}(\overline{s'}) < \mathsf{h}(\overline{s})$, hence we can apply the induction hypothesis: $l_h(s')$ is finite and thus $l_h(s) = l_h(s') + 1$ is finite.
— In the case where $t = (v)u$, we have $s \succ_h (v, e) \cdot ((u, e) \cdot \pi)$. Set $s' = (v, e) \cdot ((u, e) \cdot \pi)$. We have $\overline{s'} = \overline{s}$ and thus we can apply the induction hypothesis: $l_h(s')$ is finite and thus $l_h(s) = l_h(s') + 1$ is finite.
— In the case where $t \in \mathcal{V} \setminus \mathsf{dom}(e)$, we have $l_h(s) = 1$.
— In the case where $t = \lambda x.u$ and $\pi = \varepsilon$, we have $s \succ_h \lambda x.((u, e) \cdot \varepsilon)$. Set $s' = (u, e) \cdot \varepsilon$. Since s respects the variable convention, we have $\overline{s} = \lambda x.u[e] = \lambda x.\overline{s'}$. We have $\mathsf{h}(\overline{s'}) = \mathsf{h}(\overline{s})$, hence we can apply the induction hypothesis: $l_h(s')$ is finite and thus $l_h(s) = l_h(s') + 1$ is finite. □

We recall that if a λ-term t has a head normal form, then the last term of the terminating head reduction of t is called *the principal head normal form of t* (see Barendregt (1984)). Proposition 2.6, Lemma 2.8 and Theorem 2.10 together show that, for any head normalisable λ-term t having t' as principal head normal form, we have $(t, \bot) \cdot \varepsilon {\succ_h}^* t'$ and t' is a Krivine head normal form.

2.3. *A machine computing the β-normal form*

We now slightly modify the machine so as to compute the β-normal form of any normalisable λ-term.

Definition 2.11. We define a partial map from $\mathcal{K}$ to $\mathcal{K}$: For any $k, k' \in \mathcal{K}$, the notation $k \succ_\beta k'$ will mean that the map assigns k' to k. The value of the map at k is defined, by induction on $\mathsf{d}(k)$, as follows:

$$k \mapsto \begin{cases} s' & \text{if } k \in \mathbb{S} \text{ and } k \succ_{\mathbb{S}} s' \\ (x)(c_1 \cdot \varepsilon) \ldots (c_q \cdot \varepsilon) & \text{if } k = ((x, e), c_1, \ldots, c_q) \in \mathbb{S} \text{ and } x \in \mathcal{V} \setminus \mathsf{dom}(e) \\ \lambda x.((u, e) \cdot \varepsilon) & \text{if } k = (\lambda x.u, e) \cdot \varepsilon \in \mathbb{S} \\ (v')u & \text{if } k = (v)u \text{ and } v \succ_\beta v' \\ (v)k'_0 k_1 \ldots k_q & \text{if } k = (v)k_0 \ldots k_q,\ v \in \Lambda \cap \mathcal{H} \text{ normal and } k_0 \succ_\beta k'_0 \\ \lambda y.k'_0 & \text{if } k = \lambda y.k_0 \text{ and } k_0 \succ_\beta k'_0 \end{cases}$$

Let us compare Definition 2.11 with Definition 2.4. The difference is in the case where the current subterm of a state is a variable and where this variable has no value in the environment: The first machine stops, the second machine continues to compute every argument of the variable.

The function l_β is defined as l_h (see Definition 2.5), but for this new machine. For any $k \in \mathcal{K}$, k is said to be a *Krivine normal form* if for any $k' \in \mathcal{K}$, we do not have $k \succ_\beta k'$.

Proposition 2.12. For any $s \in \mathbb{S}$, for any Krivine normal form $k' \in \mathcal{K}$ such that $s {\succ_\beta}^* k'$, k' is a λ-term in normal form.

Proof. By induction on $l_\beta(s)$. The base case is trivial, because we never have $l_\beta(s) = 0$. The inductive step is divided into five cases.

— If $s = (x, e) \cdot (c_1, \ldots, c_q)$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, then $k' = (x)k_1 \ldots k_q$ with $c_1 \cdot \varepsilon {\succ_\beta}^* k_1$, $\ldots$, $c_q \cdot \varepsilon {\succ_\beta}^* k_q$. Now, $c_1 \cdot \varepsilon, \ldots, c_q \cdot \varepsilon \in \mathbb{S}$, hence, by induction hypothesis, $k_1, \ldots, k_q$ are λ-terms in normal form; therefore, k' too is a λ-term in normal form.

— If $s = (x, e) \cdot \pi$ and $x \in \mathsf{dom}(e)$, then $s \succ_\beta e(x) \cdot \pi$. Now, $e(x) \cdot \pi \in \mathbb{S}$ and $e(x) \cdot \pi {\succ_\beta}^* k'$, hence, by induction hypothesis, k' is a λ-term in normal form.

— If $s = (\lambda x.u, e) \cdot \varepsilon$, then $k' = \lambda x.k''$ with $(u, e) \cdot \varepsilon {\succ_\beta}^* k''$. Now, $(u, e) \cdot \varepsilon \in \mathbb{S}$, hence, by induction hypothesis, k'' is a λ-term in normal form; therefore, k' too is a λ-term in normal form.

— If $s = (\lambda x.u, e) \cdot (c \cdot \pi)$, then $s \succ_\beta (u, e \cup \{x \mapsto c\}) \cdot \pi$. Now, $(u, e \cup \{x \mapsto c\}) \cdot \pi \in \mathbb{S}$ and $(u, e \cup \{x \mapsto c\}) \cdot \pi {\succ_\beta}^* k'$, hence, by induction hypothesis, k' is a λ-term in normal form.

— If $s = ((v)u, e) \cdot \pi$, then $s \succ_\beta (v, e) \cdot ((u, e) \cdot \pi)$. Now, $(v, e) \cdot ((u, e) \cdot \pi) \in \mathbb{S}$ and $(v, e) \cdot ((u, e) \cdot \pi) {\succ_\beta}^* k'$, hence, by induction hypothesis, k' is a λ-term in normal form. □

Lemma 2.13. For any $k, k' \in \mathcal{K}$, if $k \succ_\beta k'$, then $\overline{k} \rightarrow_\ell \overline{k'}$, where $\rightarrow_\ell$ is the reflexive closure of the leftmost reduction.

Proof. By Noetherian induction on $\mathsf{d}(k)$. There are four cases:

— If $k \in \mathbb{S}$, then there are five cases:

  – If $k = (x, e) \cdot (c_1, \ldots, c_q)$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, then $\overline{k} = (x)\overline{c_1} \ldots \overline{c_q}$ and $\overline{k'} = \overline{(x)(c_1 \cdot \varepsilon) \ldots (c_q \cdot \varepsilon)} = (x)\overline{c_1} \ldots \overline{c_q}$, we have $\overline{k} = \overline{k'}$.

  – The cases where $k = (x, e) \cdot (c_1, \ldots, c_q)$ with $x \in \mathsf{dom}(e)$ or $k = (\lambda x.u, e) \cdot \varepsilon$ or $k = (\lambda x.u, e) \cdot (c_0, \ldots, c_q)$ or $k = ((v)u, e) \cdot (c_1 \ldots c_q)$ have to be dealt with in the same way as in the proof of Lemma 2.8.

— If $k = (v)u$ and $v \succ_\beta v'$, then $\overline{k} = (\overline{v})\overline{u}$ and $\overline{k'} = \overline{(v')u} = (\overline{v'})\overline{u}$. By induction hypothesis, we have $\overline{v} \rightarrow_\ell \overline{v'}$; hence, we have $\overline{k} \rightarrow_\ell \overline{k'}$.

— If $k = (v)k_0 \ldots k_q$, $v \in \Lambda \cap \mathcal{H}$ normal and $k_0 \succ_\beta k'_0$, then $\overline{k} = (v)\overline{k_0} \ldots \overline{k_q}$ and $\overline{k'} = (v)\overline{k'_0}\overline{k_1} \ldots \overline{k_q}$. By induction hypothesis, we have $\overline{k_0} \rightarrow_\ell \overline{k'_0}$; hence, we have $\overline{k} \rightarrow_\ell \overline{k'}$.

— If $k = \lambda y.k_0$, then $\overline{k} = \lambda y.\overline{k_0}$ and $\overline{k'} = \overline{\lambda y.k'_0} = \lambda y.\overline{k'_0}$ with $k_0 \succ_\beta k'_0$: By induction hypothesis, we have $\overline{k_0} \rightarrow_\ell \overline{k'_0}$, hence $\overline{k} \rightarrow_\ell \overline{k'}$. □

Theorem 2.14. For any $k \in \mathcal{K}$, if $l_\beta(k)$ is finite, then $\overline{k}$ is normalisable.

Proof. By induction on $l_\beta(k)$.
If $l_\beta(k) = 0$, then k is normal. Otherwise, apply Lemma 2.13. □

For any normalisable λ-term t, we denote by $\mathsf{n}(t)$ the number of steps leading from t to its normal form following the leftmost reduction strategy.

Theorem 2.15. For any $s = (t, e) \cdot \pi \in \mathbb{S}$, if $\overline{s}$ is normalisable, then $l_\beta(s)$ is finite.

Proof. We prove, by Noetherian induction on $(\mathsf{n}(\overline{s}), \overline{s}, d(e), t)$ with respect to the lexicographical order on $\mathbb{N} \times \Lambda \times \mathbb{N} \times \Lambda$, that for any $s = (t, e) \cdot \pi$ such that $\overline{s}$ is normalisable, $l_\beta(s)$ is finite.
If $\mathsf{n}(\overline{s}) = 0$, $\overline{s} \in \mathcal{V}$, $\mathsf{d}(e) = 0$ and $t \in \mathcal{V}$, then we have $l_\beta(s) = 1$.
Otherwise, there are five cases:

— In the case where $t \in \mathsf{dom}(e)$, we have $s \succ_\beta e(t) \cdot \pi$. Set $s' = e(t) \cdot \pi$ and $e(t) = (t', e')$. We have $\overline{s} = \overline{s'}$ and $\mathsf{d}(e') < \mathsf{d}(e)$; hence, we can apply the induction hypothesis: $l_\beta(s')$ is finite and thus $l_\beta(s) = l_\beta(s') + 1$ is finite.

— In the case where $t = \lambda x.u$ and $\pi = c \cdot \pi'$, we have $s \succ_\beta (u, e \cup \{x \mapsto c\}) \cdot \pi'$. Set $s' = (u, e \cup \{x \mapsto c\}) \cdot \pi'$. We have $\mathsf{n}(\overline{s'}) < \mathsf{n}(\overline{s})$; hence, we can apply the induction hypothesis: $l_\beta(s')$ is finite and thus $l_\beta(s) = l_\beta(s') + 1$ is finite.

— In the case where $t = (v)u$, we have $s \succ_\beta (v, e) \cdot ((u, e) \cdot \pi)$. Set $s' = (v, e) \cdot ((u, e) \cdot \pi)$. We have $\overline{s'} = \overline{s}$, and thus we can apply the induction hypothesis: $l_\beta(s')$ is finite and thus $l_\beta(s) = l_\beta(s') + 1$ is finite.

— In the case where $t \in \mathcal{V} \setminus \mathsf{dom}(e)$, let $\pi = (c_1, \ldots, c_q)$. For any $k \in \{1, \ldots, q\}$, we have $\mathsf{n}(\overline{c_k}) \leqslant \mathsf{n}(\overline{s})$ and $\overline{c_k} < \overline{s}$, hence we can apply the induction hypothesis on c_k: For any $k \in \{1, \ldots, q\}$, $l_\beta(c_k)$ is finite, hence $l_\beta(s) = \sum_{k=1}^{q} l_\beta(c_k) + 1$ is finite too.

— In the case where $t = \lambda x.u$ and $\pi = \varepsilon$, we have $s \succ_\beta \lambda x.((u,e)\cdot\varepsilon)$. Set $s' = (u,e)\cdot\varepsilon$. Since s respects the variable convention, we have $\overline{s} = \lambda x.u[e] = \lambda x.\overline{s'}$. We have $\mathsf{n}(\overline{s'}) = \mathsf{n}(\overline{s})$ and $\overline{s'} < \overline{s}$, hence we can apply the induction hypothesis: $l_\beta(s')$ is finite and thus $l_\beta(s) = l_\beta(s') + 1$ is finite. □

Proposition 2.12, Lemma 2.13 and Theorem 2.15 together show that, for any normalisable λ-term t having t' as normal form, we have $(t,\bot)\cdot\varepsilon {\succ_\beta}^* t'$ and t' is a Krivine normal form.

3. A non-uniform semantics of λ-calculus

We define here the semantics allowing to measure execution time. We have in mind the following philosophy: The semantics of the untyped λ-calculus come from the semantics of the simply typed λ-calculus and any semantics of linear logic induces a semantics of the simply typed λ-calculus. For this reason, we start from a semantics $\mathfrak{M}$ of linear logic (Subsection 3.1), then we present the induced semantics $\Lambda(\mathfrak{M})$ of the simply typed λ-calculus (Subsection 3.2) and lastly the semantics of the untyped λ-calculus that we consider (Subsection 3.3). This semantic is *non-uniform* in the sense that the interpretation of a function contains information about its behaviour on arguments whose value can change during the computation: In Subsection 3.4, we give an example illustrating this point.

Starting from a semantics of linear logic permits one to

— explain where the idea of considering the semantics of λ-calulus we consider comes from;
— give a more abstract and elegant description of the semantics of λ-calculus (as a co-Kleisli category of some comonad) and to give a more conceptual proof of Proposition 3.1;
— emphasize that this model can be approximated by a model of linear logic (this is not always the case) and by which model this can be done (even when it is the case, it is not trivial to discover it);
— relate some apparently non-related works (for instance, Coppo et al. 1980 and Tortora de Falco 2000);
— promise an adaptation of this work to linear logic (and this was carried out in de Carvalho et al. (2008)).

Formally, however, Section 3.1 is independent of what follows, so that the reader, if he wants (or if he is not acquainted with semantics of linear logic), can skip this subsection, especially if he accepts Proposition 3.1.

3.1. *A relational model of linear logic*

The first works tackling the problem of giving a general categorical definition of a denotational semantics of linear logic are those of Lafont (1988) and of Seely (1989). As for the works of Benton et al. (1994), Bierman (1993) and Bierman (1995), they led to the following axiomatic: A categorical model of the multiplicative exponential fragment of intuitionistic linear logic (IMELL) is a quadruple $(\mathcal{C}, \mathcal{L}, c, w)$ such that

— $\mathcal{C} = (\mathbb{C}, \otimes, I, \alpha, \lambda, \rho, \gamma)$ is a closed symmetric monoidal category;
— $\mathcal{L} = ((T, \mathsf{m}, \mathsf{n}), \delta, d)$ is a symmetric monoidal comonad on $\mathcal{C}$;
— c is a monoidal natural transformation from $(T, \mathsf{m}, \mathsf{n})$ to $\otimes \circ \Delta_{\mathcal{C}} \circ (T, \mathsf{m}, \mathsf{n})$, where $\Delta_{\mathcal{C}}$ is the diagonal monoidal functor from $\mathcal{C}$ to $\mathcal{C} \times \mathcal{C}$, and w is a monoidal natural transformation from $(T, \mathsf{m}, \mathsf{n})$ to $*_{\mathcal{C}}$, where $*_{\mathcal{C}}$ is the monoidal endofunctor on $\mathcal{C}$ that sends each arrow of $\mathbb{C}$ to id_I, such that
  – for any object A of $\mathbb{C}$, $((T(A), \delta_A), c_A, w_A)$ is a cocommutative comonoid in $(\mathbb{C}^{\mathbb{T}}, \otimes^{\mathbb{T}}, (I, \mathsf{n}), \alpha, \lambda, \rho)$
  – and for any $f \in \mathbb{C}^{\mathbb{T}}[(T(A), \delta_A), (T(B), \delta_B)]$, f is a comonoid morphism,

  where $\mathbb{T}$ is the comonad (T, δ, d) on $\mathbb{C}$ and $\mathbb{C}^{\mathbb{T}}$ is the category of $\mathbb{T}$-coalgebras.

Given a categorical model $\mathfrak{M} = (\mathcal{C}, \mathcal{L}, c, w)$ of IMELL with $\mathcal{C} = (\mathbb{C}, \otimes, I, \alpha, \lambda, \rho, \gamma)$ and $\mathcal{L} = ((T, \mathsf{m}, \mathsf{n}), \delta, d)$, we can define a cartesian closed category $\Lambda(\mathfrak{M})$ such that

— objects are finite sequences of objects of $\mathbb{C}$
— and arrows $(A_1, \dots, A_m) \to (B_1, \dots, B_p)$ are the sequences $(f_1, \dots, f_p)$ such that every f_k is an arrow $\bigotimes_{j=1}^{m} T(A_j) \to B_k$ in $\mathbb{C}$.

Hence, we can interpret simply typed λ-calculus in the category $\Lambda(\mathfrak{M})$. This category is (weakly) equivalent[†] to a full subcategory of (T, δ, d)-coalgebras exhibited by Hyland. If the category $\mathbb{C}$ is cartesian, then the category $\Lambda(\mathfrak{M})$ and the co-Kleisli category of the comonad (T, δ, d) are (strongly) equivalent[‡]. See de Carvalho (2007) for a full exposition.

Below, we describe completely the category $\Lambda(\mathfrak{M})$ (with its composition operation and its identities) only for the particular case that we consider in this paper.

The category of sets and relations is denoted by **Rel** and its composition operation by $\circ$. The functor T from **Rel** to **Rel** is defined by setting

— for any object A of **Rel**, $T(A) = \mathcal{M}(A)$;
— and, for any $f \in \mathbf{Rel}(A, B)$, $T(f) \in \mathbf{Rel}(T(A), T(B))$ defined by

$$T(f) = \bigcup_{n \in \mathbb{N}} \{([\alpha_1, \dots, \alpha_n], [\beta_1, \dots, \beta_n]) \ / \ (\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n) \in f\}.$$

The natural transformation d from T to the identity functor of **Rel** is defined by setting $d_A = \{([\alpha], \alpha) \ / \ \alpha \in A\}$ and the natural transformation δ from T to $T \circ T$ by setting $\delta_A = \bigcup_{n \in \mathbb{N}} \{(\sum_{i=1}^{n} a_i, [a_1, \dots, a_n]) \ / \ a_1, \dots, a_n \in T(A)\}$. It is easy to show that (T, δ, d) is a comonad on **Rel**. It is well-known that this comonad can be provided with a structure $\mathfrak{M}$ that is a denotational semantics of (I)MELL.

This denotational semantics gives rise to a cartesian closed category $\Lambda(\mathfrak{M})$.

[†] A category $\mathbb{C}$ is said to be *weakly equivalent* to a category $\mathbb{D}$ if there exists a functor $F : \mathbb{C} \to \mathbb{D}$ full and faithful such that every object D of $\mathbb{D}$ is isomorphic to $F(C)$ for some object C of $\mathbb{C}$.
[‡] A category $\mathbb{C}$ is said to be *strongly equivalent* to a category $\mathbb{D}$ if there are functors $F : \mathbb{C} \to \mathbb{D}$ and $G : \mathbb{D} \to \mathbb{C}$ and natural isomorphisms $G \circ F \cong id_{\mathbb{C}}$ and $F \circ G \cong id_{\mathbb{D}}$.

3.2. *Interpreting simply typed λ-terms*

We give the complete description of the category $\Lambda(\mathfrak{M})$ induced by the denotational semantics $\mathfrak{M}$ of (I)MELL evoked in the previous subsection:

— objects are finite sequences of sets;
— arrows $(A_1, \ldots, A_m) \to (B_1, \ldots, B_n)$ are the sequences $(f_1, \ldots, f_n)$ such that every f_i is a subset of $(\prod_{j=1}^{m} \mathcal{M}(A_j)) \times B_i$;
— if $f = (f_1, \ldots, f_p)$ is an arrow $(A_1, \ldots, A_m) \to (B_1, \ldots, B_p)$, then $f^! = \bigcup_{(n_1,\ldots,n_p)\in\mathbb{N}^p} f^{(n_1,\ldots,n_p)}$, where $f^{(n_1,\ldots,n_p)}$ is the set

$$\left\{ \begin{array}{l} ((\sum_{k=1}^{p} \sum_{i=1}^{n_k} a_1^{i,k}, \ldots, \sum_{k=1}^{p} \sum_{i=1}^{n_k} a_m^{i,k}), ([\beta_1^1, \ldots, \beta_1^{n_1}], \ldots, [\beta_p^1, \ldots, \beta_p^{n_p}])) \ / \\ \quad (\forall k \in \{1, \ldots, p\}) \, (\forall i \in \{1, \ldots, n_k\}) \, ((a_1^{i,k}, \ldots, a_m^{i,k}), \beta_k^i) \in f_k \end{array} \right\};$$

— if $f = (f_1, \ldots, f_p)$ is an arrow $(A_1, \ldots, A_m) \to (B_1, \ldots, B_p)$ and $(g_1, \ldots, g_q)$ is an arrow $(B_1, \ldots, B_p) \to (C_1, \ldots, C_q)$, then $(g_1, \ldots, g_q) \circ_{\Lambda(\mathfrak{M})} (f_1, \ldots, f_p)$ is the arrow $(h_1, \ldots, h_q) : (A_1, \ldots, A_m) \to (C_1, \ldots, C_q)$, where, for $1 \leqslant l \leqslant q$, h_l is

$$\bigcup_{b_1 \in \mathcal{M}(B_1), \ldots, b_p \in \mathcal{M}(B_p)} \left\{ \begin{array}{l} ((a_1, \ldots, a_m), \gamma) \ / \ ((b_1, \ldots, b_p), \gamma) \in g_l \text{ and} \\ \quad ((a_1, \ldots, a_m), (b_1, \ldots, b_p)) \in f^! \end{array} \right\};$$

— the identity of $(A_1, \ldots, A_m)$ is $(d^1, \ldots, d^m)$ with

$$d^j = \{((\underbrace{[], \ldots, []}_{j-1 \text{ times}}, [\alpha], \underbrace{[], \ldots, []}_{m-j \text{ times}}), \alpha) \ / \ \alpha \in A_j\}.$$

Proposition 3.1. The category $\Lambda(\mathfrak{M})$ has the following cartesian closed structure:

— for any object A, there is exactly one arrow from A to the empty sequence $()$;
— the product object and the pairing $(\ldots, \ldots)_{\mathfrak{M}}$ are concatenation;
— for any objects $B' = (B_1, \ldots, B_p)$ and $B'' = (B_{p+1}, \ldots, B_{p+q})$, the projections $\pi^1_{B',B''} : (B_1, \ldots, B_{p+q}) \to (B_1, \ldots, B_p)$ and $\pi^2_{B',B''} : (B_1, \ldots, B_{p+q}) \to (B_{p+1}, \ldots, B_{p+q})$ are defined by setting $\pi^1_{B',B''} = (d^1, \ldots, d^p)$ and $\pi^2_{B',B''} = (d^{p+1}, \ldots, d^{p+q})$ with $d^k = \{((\underbrace{[], \ldots, []}_{k-1 \text{ times}}, [\beta], \underbrace{[], \ldots, []}_{p+q-k \text{ times}}), \beta) \ / \ \beta \in B_k\}$;
— for any objects $A = (A_1, \ldots, A_m)$ and $C = (C_1, \ldots, C_q)$, the exponential object $A \Rightarrow C$ is $((\prod_{j=1}^{m} \mathcal{M}(A_j)) \times C_1, \ldots, (\prod_{j=1}^{m} \mathcal{M}(A_j)) \times C_q)$;
— for $h = (h_1, \ldots, h_q) : (A_1, \ldots, A_m, B_1, \ldots, B_p) \to (C_1, \ldots, C_q)$, the abstraction

$$\Lambda^{(B_1,\ldots,B_p)}_{(A_1,\ldots,A_m),(C_1,\ldots,C_q)}(h) : (A_1, \ldots, A_m) \to (B_1 \ldots, B_p) \Rightarrow (C_1, \ldots, C_q)$$

is defined by induction on p:

- if $p = 0$, then $\Lambda^{(B_1,\ldots,B_p)}_{(A_1,\ldots,A_m),(C_1,\ldots,C_q)}(h) = h$;
- if $p = 1$, then $\Lambda^{(B_1,\ldots,B_p)}_{(A_1,\ldots,A_m),(C_1,\ldots,C_q)}(h) = (\xi^{\mathcal{M}(B_1)}_{\prod_{j=1}^{m} \mathcal{M}(A_j),C_1}(h_1), \ldots, \xi^{\mathcal{M}(B_1)}_{\prod_{j=1}^{m} \mathcal{M}(A_j),C_q}(h_q))$ with $\xi^{\mathcal{M}(B_1)}_{\prod_{j=1}^{m} \mathcal{M}(A_j),C_l}(h_l) = \{(a_1, \ldots, a_m, (b, \gamma))/((a_1, \ldots, a_m, b), \gamma) \in h_l\}$;
- if $p \geqslant 1$, then $\Lambda^{(B_1,\ldots,B_{p+1})}_{A,C}(h) = \Lambda^{(B_1,\ldots,B_p)}_{A,(\mathcal{M}(B_{p+1})\times C_1,\ldots,\mathcal{M}(B_{p+1})\times C_q)}(\Lambda^{(B_{p+1})}_{(A_1,\ldots,A_m,B_1,\ldots,B_p),C}(h))$ with $A = (A_1, \ldots, A_m)$ and $C = (C_1, \ldots, C_q)$;

— the evaluation $\mathrm{ev}_{(C_1,\dots,C_q),(B_1,\dots,B_p)}$ is an arrow from the concatenation of $(B_1,\dots,B_p) \Rightarrow (C_1,\dots,C_q)$ and $(B_1,\dots,B_p)$ to $(C_1,\dots,C_q)$ defined by setting

$$\mathrm{ev}_{(C_1,\dots,C_q),(B_1,\dots,B_p)} = (\mathrm{ev}^1_{(C_1,\dots,C_q),(B_1,\dots,B_p)},\dots,\mathrm{ev}^q_{(C_1,\dots,C_q),(B_1,\dots,B_p)}),$$

where for $1 \leqslant k \leqslant q$, $\mathrm{ev}^k_{(C_1,\dots,C_q),(B_1,\dots,B_p)}$ is the set

$$\left\{ \begin{array}{l} ((\underbrace{[],\dots,[]}_{k-1 \text{ times}}, [((b_1,\dots,b_p),\gamma)], \underbrace{[],\dots,[]}_{q-k \text{ times}}, b_1,\dots,b_p),\gamma) \; / \\ b_1 \in \mathcal{M}(B_1),\dots,b_p \in \mathcal{M}(B_p), \gamma \in C_k \end{array} \right\}.$$

Proof. By checking some computations; or by applying Theorem 3.3.46 of de Carvalho (2007), that states that, for any denotational semantics $\mathfrak{M} = ((\mathbb{C},\dots),((T,\dots),\delta,d),\dots)$ of IMELL, if the category $\mathbb{C}$ is cartesian, then the category $\Lambda(\mathfrak{M})$ and the co-Kleisli category of the comonad (T,δ,d) are equivalent, and Seely's Proposition, that states that the latter is cartesian closed. □

3.3. *Interpreting type-free λ-terms*

First, we recall that if $f : D \to C$ and $g : C \to D$ are two arrows in a category $\mathbb{C}$, then f *is a retraction of* g *in* $\mathbb{C}$ means that $f \circ_{\mathbb{C}} g = id_C$ (see, for instance, Mac Lane 1998); it is also said that (g,f) *is a retraction pair.*

With the cartesian closed structure on $\Lambda(\mathfrak{M})$, we have a semantics of the simply typed λ-calculus (see, for instance, Lambek and Scott 1986). Now, in order to have a semantics of the pure λ-calculus, it is therefore enough to have a *reflexive* object U of $\Lambda(\mathfrak{M})$, that is to say such that $(U \Rightarrow U) \lhd U$, that means that there exist $s \in \Lambda(\mathfrak{M})[U \Rightarrow U, U]$ and $r \in \Lambda(\mathfrak{M})[U, U \Rightarrow U]$ such that $r \circ_{\Lambda(\mathfrak{M})} s$ is the identity on $U \Rightarrow U$; in particular, (s,r) is a retraction pair. We will use the following lemma for exhibiting such a retraction pair.

Lemma 3.2. Let $h : A \to B$ be an injection between sets. Consider the arrows $g : \mathcal{M}(A) \to B$ and $f : \mathcal{M}(B) \to A$ of the category **Rel** defined by $g = \{([\alpha], h(\alpha))/\alpha \in A\}$ and $f = \{([h(\alpha)],\alpha)/\alpha \in A\}$. Then, $(g) \in \Lambda(\mathfrak{M})((A),(B))$ and (f) is a retraction of (g) in $\Lambda(\mathfrak{M})$.

Proof. We have

$$(g)^! = \bigcup_{n\in\mathbb{N}} \{([\alpha_1,\dots,\alpha_n],[h(\alpha_1),\dots,h(\alpha_n)]) \; / \; (\forall i \in \{1,\dots,n\})\, \alpha_i \in A\}.$$

Hence, $(f) \circ_{\Lambda(\mathfrak{M})} (g) = id_{(A)}$. □

If D is a set, then $(D) \Rightarrow (D) = (\mathcal{M}(D) \times D)$. From now on, we assume that D is a non-empty set and that h is an injection from $\mathcal{M}(D) \times D$ to D.

Set $g = \{([\alpha], h(\alpha)) \; / \; \alpha \in \mathcal{M}(D) \times D\} : \mathcal{M}(\mathcal{M}(D) \times D) \to D$ in **Rel**
and $f = \{([h(\alpha)],\alpha) \; / \; \alpha \in \mathcal{M}(D) \times D\} : \mathcal{M}(D) \to \mathcal{M}(D) \times D$ in **Rel**.

Lemma 3.2 shows that we have $((D) \Rightarrow (D)) \lhd (D)$ in the category $\Lambda(\mathfrak{M})$ and, more precisely, $(g) \in \Lambda(\mathfrak{M})((D) \Rightarrow (D),(D))$ and (f) is a retraction of (g).

We can therefore define the interpretation of any λ-term.

Definition 3.3. For any λ-term t possibly containing constants from $\mathcal{P}(D)$, for any $x_1, \ldots, x_m \in \mathcal{V}$ distinct such that $FV(t) \subseteq \{x_1, \ldots, x_m\}$, we define, by induction on t, $[\![t]\!]_{x_1,\ldots,x_m} \subseteq (\prod_{j=1}^{m} \mathcal{M}(D)) \times D$:

— $[\![x_j]\!]_{x_1,\ldots,x_m} = \{((\underbrace{[], \ldots, []}_{j-1 \text{ times}}, [\alpha], \underbrace{[], \ldots, []}_{m-j \text{ times}}), \alpha) \mathbin{/} \alpha \in D\}$;

— for any $c \in \mathcal{P}(D)$, $[\![c]\!]_{x_1,\ldots,x_m} = \{((\underbrace{[], \ldots, []}_{m \text{ times}}), \alpha) \mathbin{/} \alpha \in c\}$;

— $[\![\lambda x.u]\!]_{x_1,\ldots,x_m} = \{((a_1, \ldots, a_m), h(a, \alpha)) \mathbin{/} ((a_1, \ldots, a_m, a), \alpha) \in [\![u]\!]_{x_1,\ldots,x_m,x}\}$;

— the value of $[\![(v)u]\!]_{x_1,\ldots,x_m}$ is

$$\bigcup_{n \in \mathbb{N}} \bigcup_{\alpha_1,\ldots,\alpha_n \in D} \left\{ \begin{array}{ll} ((\sum_{i=0}^{n} a_1^i, \ldots, \sum_{i=0}^{n} a_m^i), \alpha) \mathbin{/} & \\ & ((a_1^0, \ldots, a_m^0), h([\alpha_1, \ldots, \alpha_n], \alpha)) \in [\![v]\!]_{x_1,\ldots,x_m} \\ \text{and} & (\forall i \in \{1, \ldots, n\})((a_1^i, \ldots, a_m^i), \alpha_i) \in [\![u]\!]_{x_1,\ldots,x_m} \end{array} \right\}.$$

Now, we can define the interpretation of any λ-term in any environment.

Definition 3.4. For any $\rho \in \mathcal{P}(D)^{\mathcal{V}}$ and for any λ-term t possibly containing constants from $\mathcal{P}(D)$ such that $FV(t) = \{x_1, \ldots, x_m\}$, we set

$$[\![t]\!]_\rho = \bigcup_{a_1 \in \mathcal{M}(\rho(x_1)),\ldots,a_m \in \mathcal{M}(\rho(x_m))} \{\alpha \in D \mathbin{/} ((a_1, \ldots, a_m), \alpha) \in [\![t]\!]_{x_1,\ldots,x_m}\}.$$

Definition 3.5. For any $d_1, d_2 \in \mathcal{P}(D)$, we set

$$d_1 * d_2 = \bigcup_{a \in \mathcal{M}(d_2)} \{\alpha \in D \mathbin{/} h(a, \alpha) \in d_1\}.$$

We have

Proposition 3.6. The triple $(\mathcal{P}(D), *, [\![-]\!]_-)$ is a λ-algebra.

Proof. Theorem 5.5.6 of Barendregt (1984) shows that we obtain a λ-algebra if we have a cartesian closed structure as in Proposition 3.1 and a retraction pair (g, f) as defined above, and if we set

— $[\![t]\!]_\rho = ([\![t]\!]_{x_1,\ldots,x_m}) \circ_{\Lambda(\mathfrak{M})} ((\rho(x_1)), \ldots, (\rho(x_m)))_{\mathfrak{M}}$, where

- $\{x_1, \ldots, x_m\} = FV(t)$;
- $$((\rho(x_1)), \ldots, (\rho(x_m)))_{\mathfrak{M}} = \left\{ \begin{array}{l} id_{()} \text{ if } m = 0; \\ (\rho(x_1)) : () \to (D) \text{ if } m = 1; \\ (((\rho(x_1)), \ldots, (\rho(x_{m-1})))_{\mathfrak{M}}, (\rho(x_m)))_{\mathfrak{M}} \text{ if } m \geqslant 3. \end{array} \right.$$
- $([\![t]\!]_{x_1,\ldots,x_m})$ is the morphism $\underbrace{(D, \ldots, D)}_{m \text{ times}} \to (D)$ in the category $\Lambda(\mathfrak{M})$ defined as in Definition 3.3;

— and $(d_1 * d_2) = ev_{(D),(D)} \circ_{\Lambda(\mathfrak{M})} ((f) \circ_{\Lambda(\mathfrak{M})} (d_1), (d_2))_{\mathfrak{M}}$.

Straightforward computations show that we obtain Definitions 3.4 and 3.5, note in particular that

— $((\rho(x_1)),\dots,(\rho(x_m)))_{\mathfrak{M}} = (\rho(x_1),\dots,\rho(x_m)) : () \to \underbrace{(D,\dots,D)}_{m \text{ times}}$ and

$$(\rho(x_1),\dots,\rho(x_m))^! = \begin{cases} \{()\} & \text{if } m = 0; \\ \prod_{j=1}^{m} \mathcal{M}(\rho(x_j)) & \text{if } m \neq 0; \end{cases}$$

— $ev_{(D),(D)} = (ev^1_{(D),(D)}) : (D \Rightarrow D, D) \to (D)$ with $ev^1_{(D),(D)} = \{(([(a,\alpha)],a),\alpha) \ / \ a \in \mathcal{M}(D)$ and $\alpha \in D\}$;

— $(f) \circ_{\Lambda(\mathfrak{M})} (d_1) = (l) : () \to (D \Rightarrow D)$ with $l = \{(a,\alpha) \in \mathcal{M}(D) \times D \ / \ h(a,\alpha) \in d_1\}$. □

But the following proposition states that the triple $(\mathcal{P}(D), *, [\![-]\!]_-)$ is *not* a λ-model. We recall (see, for instance, Barendregt 1984), that a λ-model is a λ-algebra $(\mathcal{D}, *, [\![-]\!]_-)$ such that the following property, expressing the ξ-rule, holds:

For any $\rho \in \mathcal{D}^{\mathcal{V}}$, for any $x \in \mathcal{V}$ and for any λ-terms t_1 and t_2, we have

$$((\forall d \in \mathcal{D})[\![t_1]\!]_{\rho[x:=d]} = [\![t_2]\!]_{\rho[x:=d]} \Rightarrow [\![\lambda x.t_1]\!]_\rho = [\![\lambda x.t_2]\!]_\rho).$$

Proposition 3.7. The λ-algebra $(\mathcal{P}(D), *, [\![-]\!]_-)$ is not a λ-model.

In other words, there exist $\rho \in \mathcal{P}(D)^{\mathcal{V}}$, $x \in \mathcal{V}$ and two λ-terms t_1 and t_2 such that

$$((\forall d \in \mathcal{P}(D))[\![t_1]\!]_{\rho[x:=d]} = [\![t_2]\!]_{\rho[x:=d]} \text{ and } [\![\lambda x.t_1]\!]_\rho \neq [\![\lambda x.t_2]\!]_\rho).$$

In particular, $[\![t]\!]_\rho$ *cannot be* defined by induction on t (an interpretation by polynomials (see Selinger 2002) and an interpretation by 'finitary morphisms' (see Bucciarelli et al. 2007) are nevertheless possible in such a way that the ξ-rule holds).

Proof. It is enough to notice that for any non-empty set A, the object (A) does not have enough points in $\Lambda(\mathfrak{M})$. We recall that any object A of any category $\mathbb{K}$ with a terminal object is said *to have enough points* if for any terminal object 1 of $\mathbb{K}$ and for any $y, z \in \mathbb{K}(A,A)$, we have $((\forall x \in \mathbb{K}(1,A))y \circ_{\mathbb{K}} x = z \circ_{\mathbb{K}} x \Rightarrow y = z)$. Remark that it does not follow necessarily that the same holds for any $y, z \in \mathbb{K}(A,B)$.

Now, let A be a non-empty set and let $\alpha \in A$. Let y and z be the arrows $\mathcal{M}(A) \to A$ of the category **Rel** defined by $y = \{([\alpha],\alpha)\}$ and $z = \{([\alpha,\alpha],\alpha)\}$. Then, (y) and (z) are two arrows $(A) \to (A)$ of the category $\Lambda(\mathfrak{M})$.

We recall that the terminal object in $\Lambda(\mathfrak{M})$ is the empty sequence $()$. And, for any arrow $(x) : () \to (A)$ of the category $\Lambda(\mathfrak{M})$, we have $(y) \circ_{\Lambda(\mathfrak{M})} (x) = (z) \circ_{\Lambda(\mathfrak{M})} (x)$. □

This proof explains *why* Proposition 3.7 holds. A more direct proof can be obtained by considering the two λ-terms $t_1 = (y)x$ and $t_2 = (z)x$ with $\rho(y) = \{h([\alpha],\alpha)\}$ and $\rho(z) = \{h([\alpha,\alpha],\alpha)\}$ for some $\alpha \in D$.

3.4. *Non-uniformity*

Example 3.8 illustrates the non-uniformity of the semantics. It is based on the following idea.

Consider the program

```
λx.if x  then 1
         else if x  then 1
                    else 0
```

applied to a boolean. The second then is never read. A *uniform* semantics would ignore it. It is not the case when the semantics is *non-uniform*.

Example 3.8. Set $\mathbf{0} = \lambda x.\lambda y.y$ and $\mathbf{1} = \lambda x.\lambda y.x$. Assume that h is the inclusion from $\mathcal{M}(D) \times D$ to D.

Let $\gamma \in D$; set $\delta = ([], ([\gamma], \gamma))$ and $\beta = ([\gamma], ([], \gamma))$. We have

— $([([], ([\delta], \delta))], ([\delta], \delta)) \in [\![(x)\mathbf{1}]\!]_x$;
— and $([([], ([\delta], \delta))], \delta) \in [\![(x)\mathbf{10}]\!]_x$, since $([], \delta) \in [\![\mathbf{0}]\!]_x$.

Hence, we have $\alpha_1 = ([([], ([\delta], \delta)), ([], ([\delta], \delta))], \delta) \in [\![\lambda x.(x)\mathbf{1}(x)\mathbf{10}]\!]$.

We have

— $([([], ([\beta], \beta))], ([\beta], \beta)) \in [\![(x)\mathbf{1}]\!]_x$;
— and $([([\beta], ([], \beta))], \beta) \in [\![(x)\mathbf{10}]\!]_x$, since $([], \beta) \in [\![\mathbf{1}]\!]_x$.

Hence, we have $\alpha_2 = ([([], ([\beta], \beta)), ([\beta], ([], \beta))], \beta) \in [\![\lambda x.(x)\mathbf{1}(x)\mathbf{10}]\!]$.

In a uniform semantics (as in Girard 1986), the point α_1 would appear in the semantics of this λ-term, but not the point α_2, because $[([], ([\beta], \beta)), ([\beta], ([], \beta))]$ corresponds to a 'chimerical' argument: The argument is read twice and provides two contradictory values.

4. Non-idempotent intersection types

From now on, $D = \bigcup_{n \in \mathbb{N}} D_n$, where D_n is defined by induction on n: D_0 is a non-empty set A that does not contain any pairs and $D_{n+1} = A \cup (\mathcal{M}(D_n) \times D_n)$. We have $D = A \uplus (\mathcal{M}(D) \times D)$, where $\uplus$ is the disjoint union; the injection h from $\mathcal{M}(D) \times D$ to D will be the inclusion. Hence, any element of D can be written $a_1 \dots a_m\alpha$, where $a_1, \dots, a_m \in \mathcal{M}(D)$, $\alpha \in D$ and $a_1 \dots a_m\alpha$ is defined by induction on m:

— $a_1 \dots a_0\alpha = \alpha$;
— $a_1 \dots a_{m+1}\alpha = a_1 \dots a_m(a_{m+1}, \alpha)$.

For any $\alpha \in D$, we denote by $\mathsf{depth}(\alpha)$ the least integer n such that $\alpha \in D_n$.

In the preceding section, we defined the semantics we consider (Definitions 3.3 and 3.4). Now, we want to describe this semantics as a logical system: The elements of D are viewed as propositional formulas. More precisely, a comma separating a multiset of types and a type is understood as an arrow and a non-empty multiset is understood as the conjunction of its elements (their intersection). Note that this means we are considering a commutative (but not necessarily idempotent) intersection.

4.1. *System R*

A *context* Γ is a function from $\mathcal{V}$ to $\mathcal{M}(D)$ such that $\{x \in \mathcal{V} \ / \ \Gamma(x) \neq []\}$ is finite. If $x_1, \dots, x_m \in \mathcal{V}$ are distinct and $a_1, \dots, a_m \in \mathcal{M}(D)$, then $x_1 : a_1, \dots, x_m : a_m$ denotes the context defined by $x \mapsto \begin{cases} a_j & \text{if } x = x_j; \\ [] & \text{else.} \end{cases}$ We denote by Φ the set of contexts. We define the following binary operation on Φ:

$$\begin{array}{rcl} \Phi \times \Phi & \to & \Phi, \\ (\Gamma_1, \Gamma_2) & \mapsto & \Gamma_1 + \Gamma_2 : \begin{array}{rcl} \mathcal{V} & \to & \mathcal{M}(D) \\ x & \mapsto & \Gamma_1(x) + \Gamma_2(x), \end{array} \end{array}$$

where the second $+$ denotes the sum of multisets given by term-by-term addition of multiplicities. Note that this operation is associative and commutative. Typing rules concern judgements of the form $\Gamma \vdash_R t : \alpha$, where $\Gamma \in \Phi$, t is a λ-term and $\alpha \in D$.

Definition 4.1. The typing rules of System R are the following ones:

$$\frac{}{x : [\alpha] \vdash_R x : \alpha}\ ,$$

$$\frac{\Gamma, x : a \vdash_R v : \alpha}{\Gamma \vdash_R \lambda x.v : (a, \alpha)}\ ,$$

$$\frac{\Gamma_0 \vdash_R v : ([\alpha_1, \ldots, \alpha_n], \alpha) \qquad \Gamma_1 \vdash_R u : \alpha_1, \ldots, \Gamma_n \vdash_R u : \alpha_n}{\Gamma_0 + \Gamma_1 + \ldots + \Gamma_n \vdash_R (v)u : \alpha}\ n \in \mathbb{N}\ .$$

The typing rule of the application has $n+1$ premises. In particular, in the case where $n = 0$, we obtain the following rule: $\dfrac{\Gamma_0 \vdash_R v : ([], \alpha)}{\Gamma_0 \vdash_R (v)u : \alpha}$ for any λ-term u. So, the empty multiset plays the role of the universal type Ω.

The intersection we consider is *not* idempotent in the following sense: If a closed λ-term t has the type $a_1 \ldots a_m \alpha$ and, for $1 \leqslant j \leqslant m$, $\mathsf{Supp}(a'_j) = \mathsf{Supp}(a_j)$, it does not follow necessarily that t has the type $a'_1 \ldots a'_m \alpha$. For instance, the λ-term $\lambda z.\lambda x.(z)x$ has types $([([\alpha], \alpha)], ([\alpha], \alpha))$ and $([([\alpha, \alpha], \alpha)], ([\alpha, \alpha], \alpha))$ but not the type $([([\alpha], \alpha)], ([\alpha, \alpha], \alpha))$. On the contrary, the system presented in Ronchi Della Rocca (1988) and the System $\mathcal{D}$ presented in Krivine (1990) consider an idempotent intersection. System λ of Kfoury (2000) and System $\mathbb{I}$ of Neergaard and Mairson (2004) consider a non-idempotent intersection, but the treatment of weakening is not the same.

Interestingly, System R can be seen as a reformulation of the system of Coppo et al. (1980). More precisely, types of System R correspond to their normalised types.

4.2. *Relating types and semantics*

We prove in this subsection that the semantics of a closed λ-term as defined in Subsection 3.3 is the set of its types in System R. The following assertions relate more precisely types and semantics of any λ-term.

Theorem 4.2. For any λ-term t such that $FV(t) \subseteq \{x_1, \ldots, x_m\}$, we have

$$[\![t]\!]_{x_1,\ldots,x_m} = \{a_1 \ldots a_m \alpha \in \left(\prod_{j=1}^{m} \mathcal{M}(D)\right) \times D \ /\ x_1 : a_1, \ldots, x_m : a_m \vdash_R t : \alpha\}.$$

Proof. By induction on t. □

This theorem has the three following corollaries.

Corollary 4.3. For any λ-terms t and t' such that $t =_\beta t'$, if $\Gamma \vdash_R t : \alpha$, then we have $\Gamma \vdash_R t' : \alpha$.

Proof. By our Proposition 3.1 and Lemma 3.2, and Proposition 5.5.5 of Barendregt (1984), the following property holds: For any λ-terms t and t' such that $t =_\beta t'$ and such that $FV(t) \subseteq \{x_1, \ldots, x_m\}$, we have $[\![t]\!]_{x_1,\ldots,x_m} = [\![t']\!]_{x_1,\ldots,x_m}$. □

Corollary 4.4. For any λ-term t and for any $\Gamma \in \Phi$, we have

$$\begin{aligned} &\{\alpha \in D \;/\; \Gamma \vdash_R t : \alpha\} \\ \subseteq\; &\{\alpha \in D \;/\; (\forall \rho \in \mathcal{P}(D)^{\mathcal{V}})((\forall x \in \mathcal{V})\Gamma(x) \in \mathcal{M}(\rho(x)) \Rightarrow \alpha \in [\![t]\!]_\rho)\}. \end{aligned}$$

Remark 4.5. The reverse inclusion is not true.

Corollary 4.6. For any λ-term t and for any $\rho \in \mathcal{P}(D)^{\mathcal{V}}$, we have

$$[\![t]\!]_\rho = \{\alpha \in D \;/\; (\exists \Gamma \in \Phi)((\forall x \in \mathcal{V})\Gamma(x) \in \mathcal{M}(\rho(x)) \text{ and } \Gamma \vdash_R t : \alpha)\} \;.$$

There is another way to compute the interpretation of λ-terms in this semantics. Indeed, it is well-known that the translation of simply typed λ-terms into linear logic through the encoding $A \Rightarrow B \equiv ?A^\perp \wp B$ can be extended to a translation of type-free λ-terms into linear logic nets labelled with the types I, O, $?I$ and $!O$ (as in Regnier 1992). Now, we can do *experiments* (in the sense of Girard (1987), that introduced this notion in the framework of coherent semantics for working with proof-nets directly, without sequentialising) to compute the semantics of the net in the multiset-based relational model. And this semantics is the same as the semantics defined here.

For a survey of translations of λ-terms in proof-nets, see Guerrini (2004).

4.3. *An equivalence relation on derivations*

Definition 4.8 introduces an equivalence relation on the set of derivations of a given λ-term. This relation, as well as the notion of substitution defined immediately after, will play a role in Subsection 6.5.

Definition 4.7. For any λ-term t, for any $(\Gamma, \alpha) \in \Phi \times D$, we denote by $\Delta(t, (\Gamma, \alpha))$ the set of derivations of $\Gamma \vdash_R t : \alpha$.

For any λ-term t, we set $\Delta(t) = \bigcup_{(\Gamma,\alpha)\in\Phi\times D} \Delta(t, (\Gamma, \alpha))$.

For any closed λ-term t, for any $\alpha \in D$, we denote by $\Delta(t, \alpha)$ the set of derivations of $\vdash_R t : \alpha$.

For any closed λ-term t, for any integer n, for any $a \in \mathcal{M}_n(D)$, we set

$$\Delta(t, a) = \bigcup_{(\alpha_1,\ldots,\alpha_n)\in\mathfrak{S}(a)} \{(\Pi_1, \ldots, \Pi_n) \in \Delta(t)^n \;/\; (\forall i \in \{1, \ldots, n\})\, \Pi_i \in \Delta(t, \alpha_i)\} \;.$$

We set $\Delta = \bigcup_{t\in\Lambda} \Delta(t)$.

Definition 4.8. For any λ-term t, for any $\Pi, \Pi' \in \Delta(t)$, we define, by induction on t, when $\Pi \sim \Pi'$ holds:

— if t is a variable, then $\Pi \sim \Pi'$;

— if $\Pi = \dfrac{\begin{matrix}\Pi_0 \\ \Gamma, x : a \vdash_R v : \alpha\end{matrix}}{\Gamma \vdash_R \lambda x.v : (a, \alpha)}$, then $\Pi \sim \Pi'$ if and only if there exists $\Pi'_0 \sim \Pi_0$ such that

$\Pi' = \dfrac{\begin{matrix}\Pi'_0 \\ \Gamma', x : a' \vdash_R v : \alpha'\end{matrix}}{\Gamma' \vdash_R \lambda x.v : (a', \alpha')}$;

— if

$$\Pi = \dfrac{\begin{matrix}\Pi_0 \\ \Gamma_0 \vdash_R v : ([\alpha_1, \ldots, \alpha_n], \alpha)\end{matrix} \qquad \begin{matrix}\Pi_1 \\ \Gamma_1 \vdash_R u : \alpha_1\end{matrix} \quad \begin{matrix}\ldots \\ \ldots\end{matrix} \quad \begin{matrix}\Pi_n \\ \Gamma_n \vdash u : \alpha_n\end{matrix}}{\Gamma_0 + \Gamma_1 + \ldots + \Gamma_n \vdash_R (v)u : \alpha},$$

then $\Pi \sim \Pi'$ if and only if there exist $\Pi_0 \sim \Pi'_0$, $\sigma \in \mathfrak{S}_n$, $\Pi_1 \sim \Pi'_{\sigma(1)}, \ldots, \Pi_n \sim \Pi'_{\sigma(n)}$ such that

$$\Pi' = \dfrac{\begin{matrix}\Pi'_0 \\ \Gamma'_0 \vdash_R v : ([\alpha'_1, \ldots, \alpha'_n], \alpha')\end{matrix} \qquad \begin{matrix}\Pi'_1 \\ \Gamma'_1 \vdash_R u : \alpha'_1\end{matrix} \quad \begin{matrix}\ldots \\ \ldots\end{matrix} \quad \begin{matrix}\Pi'_n \\ \Gamma'_n \vdash u : \alpha'_n\end{matrix}}{\Gamma'_0 + \Gamma'_1 + \ldots + \Gamma'_n \vdash_R (v)u : \alpha'},.$$

An equivalence class of derivations of a λ-term t in System R can be seen as a *simple resource term of the shape of* t that does not reduce to 0. *Resource λ-calculus* is introduced in Ehrhard and Regnier (2006a) and is similar to resource-oriented versions of the λ-calculus previously introduced and studied in Boudol *et al.* (1999) and Kfoury (2000). For a full exposition of a precise relation between this equivalence relation and simple resource terms, see de Carvalho (2007).

Definition 4.9. A substitution σ is a function from D to D such that

$$\text{for any } \alpha, \alpha_1, \ldots, \alpha_n \in D,\ \sigma([\alpha_1, \ldots, \alpha_n], \alpha) = ([\sigma(\alpha_1), \ldots, \sigma(\alpha_n)], \sigma(\alpha)) \ .$$

We denote by $\mathcal{S}$ the set of substitutions.

For any $\sigma \in \mathcal{S}$, we denote by $\overline{\sigma}$ the function from $\mathcal{M}(D)$ to $\mathcal{M}(D)$ defined by $\overline{\sigma}([\alpha_1, \ldots, \alpha_n]) = [\sigma(\alpha_1), \ldots, \sigma(\alpha_n)]$.

Proposition 4.10. Let Π be a derivation of $\Gamma \vdash_R t : \alpha$ and let σ be a substitution. Then, there exists a derivation Π' of $\overline{\sigma} \circ \Gamma \vdash_R t : \sigma(\alpha)$ such that $\Pi \sim \Pi'$.

Proof. By induction on t. □

5. Qualitative results

In this section, inspired by Krivine (1990), we prove Theorem 5.6, which formulates *qualitative* relations between assignable types and normalisation properties: It characterises the (head) normalisable λ-terms by semantic means. We also answer to the following question: If v and u are two closed normal λ-terms, is it the case that $(v)u$ is (head) normalisable?

The answer is given only referring to $[\![v]\!]$ and $[\![u]\!]$ in Corollary 5.7. Quantitative versions of this last result will be proved in Section 6.

Definition 5.1. For any $n \in \mathbb{N}$, we define, by induction on n, D_n^{exa} and $\overline{D_n^{\mathsf{exa}}}$:

— $D_0^{\mathsf{exa}} = \overline{D_0^{\mathsf{exa}}} = A$;
— $D_{n+1}^{\mathsf{exa}} = A \cup (\mathcal{M}(\overline{D_n^{\mathsf{exa}}}) \times D_n^{\mathsf{exa}})$ and $\overline{D_{n+1}^{\mathsf{exa}}} = A \cup ((\mathcal{M}(D_n^{\mathsf{exa}}) \setminus \{[]\}) \times \overline{D_n^{\mathsf{exa}}})$.

We set

— $D^{\mathsf{exa}} = \bigcup_{n\in\mathbb{N}} D_n^{\mathsf{exa}}$;
— $\overline{D^{\mathsf{exa}}} = \bigcup_{n\in\mathbb{N}} \overline{D_n^{\mathsf{exa}}}$;
— and $\Phi^{\mathsf{exa}} = \{\Gamma \in \Phi \,/\, (\forall x \in \mathcal{V})\Gamma(x) \in \mathcal{M}(\overline{D^{\mathsf{exa}}})\}$.

Note that D^{exa} is the set of the $\alpha \in D$ such that $[]$ has no positive occurrences in α in the following sense:

— the unique occurrence of a in a is said to be positive in a;
— the positive (resp. negative) occurrences of any multiset in α are said to be positive (resp. negative) in (a, α);
— the positive (resp. negative) occurrences of any multiset in a are said to be negative (resp. positive) in (a, α).

Proposition 5.2.

i. Every head-normalisable λ-term is typable in System R.
ii. For any normalisable λ-term t, there exists $(\Gamma, \alpha) \in \Phi^{\mathsf{exa}} \times D^{\mathsf{exa}}$ such that $\Gamma \vdash_R t : \alpha$.

Proof.

i. Let t be a head-normalisable λ-term. There exists a λ-term of the shape $(\lambda x_1.\ldots\lambda x_k.t)v_1\ldots v_n$ such that $(\lambda x_1.\ldots\lambda x_k.t)v_1\ldots v_n =_\beta x$. Now, x is typable. Therefore, by Corollary 4.3, the λ-term $(\lambda x_1.\ldots\lambda x_k.t)v_1\ldots v_n$ is typable. Hence, t is typable.
ii. We prove, by induction on t, that for any normal λ-term t, the following properties hold:
   — There exists $(\Gamma, \alpha) \in \Phi^{\mathsf{exa}} \times D^{\mathsf{exa}}$ such that $\Gamma \vdash_R t : \alpha$.
   — If, moreover, t does not begin with λ, then, for any $\alpha \in D^{\mathsf{exa}}$, there exists $\Gamma \in \Phi^{\mathsf{exa}}$ such that $\Gamma \vdash_R t : \alpha$.

   Next, just apply Corollary 4.3. □

If $\mathcal{X}_1$ and $\mathcal{X}_2$ are two sets of λ-terms, then $\mathcal{X}_1 \to \mathcal{X}_2$ denotes the set of λ-terms v such that for any $u \in \mathcal{X}_1$, we have $(v)u \in \mathcal{X}_2$. A set $\mathcal{X}$ of λ-terms is said to be *saturated* if for any λ-terms $t_1, \ldots, t_n, u$ and for any $x \in \mathcal{V}$, we have

$$(u[t/x])t_1\ldots t_n \in \mathcal{X} \Rightarrow (\lambda x.u)tt_1\ldots t_n \in \mathcal{X}.$$

An interpretation is a map from A to the set of saturated sets. For any interpretation $\mathcal{I}$ and for any $\delta \in D \cup \mathcal{M}(D)$, we define, by induction on δ, a saturated set $|\delta|_{\mathcal{I}}$:

— if $\delta \in A$, then $|\delta|_{\mathcal{I}} = \mathcal{I}(\delta)$;
— if $\delta = []$, then $|\delta|_{\mathcal{I}}$ is the set of all λ-terms;

— if $\delta = [\alpha_1, \ldots, \alpha_{n+1}]$, then $|\delta|_{\mathcal{I}} = \bigcap_{i=1}^{n+1} |\alpha_i|_{\mathcal{I}}$.
— if $\delta = (a, \alpha)$, then $|\delta|_{\mathcal{I}} = |a|_{\mathcal{I}} \to |\alpha|_{\mathcal{I}}$.

Lemma 5.3. Let $\mathcal{I}$ be an interpretation and let u be a λ-term such that $x_1 : a_1, \ldots, x_k : a_k \vdash_R u : \alpha$. If $t_1 \in |a_1|_{\mathcal{I}}, \ldots, t_k \in |a_k|_{\mathcal{I}}$, then $u[t_1/x_1, \ldots, t_k/x_k] \in |\alpha|_{\mathcal{I}}$.

Proof. By induction on u. □

Lemma 5.4.

i. Let $\mathcal{N}$ be the set of head-normalisable terms. For any $\gamma \in A$, we set $\mathcal{I}(\gamma) = \mathcal{N}$. Then, for any $\alpha \in D$, we have $\mathcal{V} \subseteq |\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$.
ii. Let $\mathcal{N}$ be the set of normalisable terms. For any $\gamma \in A$, we set $\mathcal{I}(\gamma) = \mathcal{N}$. For any $\alpha \in \overline{D^{\mathsf{exa}}}$ (resp. $\alpha \in D^{\mathsf{exa}}$), we have $\mathcal{V} \subseteq |\alpha|_{\mathcal{I}}$ (resp. $|\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$).

Proof.

i. Set $\mathcal{N}_0 = \{(x)t_1 \ldots t_n \,/\, x \in \mathcal{V} \text{ and } t_1, \ldots, t_n \in \Lambda\}$. We prove, by induction on α, that we have $\mathcal{N}_0 \subseteq |\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$.
If $\alpha = (b, \beta)$, then, by induction hypothesis, we have $\mathcal{N}_0 \subseteq |\beta|_{\mathcal{I}} \subseteq \mathcal{N}$ and $\mathcal{N}_0 \subseteq |b|_{\mathcal{I}}$. Hence, we have $\mathcal{N}_0 \subseteq \Lambda \to \mathcal{N}_0 \subseteq |\alpha|_{\mathcal{I}}$ and $|\alpha|_{\mathcal{I}} \subseteq \mathcal{N}_0 \to \mathcal{N} \subseteq \mathcal{N}$.
ii. Set $\mathcal{N}_0 = \{(x)t_1 \ldots t_n \,/\, x \in \mathcal{V} \text{ and } t_1, \ldots, t_n \in \mathcal{N}\}$. We prove, by induction on α, that
— if $\alpha \in \overline{D^{\mathsf{exa}}}$, then we have $\mathcal{N}_0 \subseteq |\alpha|_{\mathcal{I}}$;
— if $\alpha \in D^{\mathsf{exa}}$, then we have $|\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$.
Suppose $\alpha = (b, \beta) \in \mathcal{M}(D) \times D$.
— If $\alpha \in \overline{D^{\mathsf{exa}}}$, then $b \in \mathcal{M}(D^{\mathsf{exa}})$ and $\beta \in \overline{D^{\mathsf{exa}}}$. By induction hypothesis, we have $|b|_{\mathcal{I}} \subseteq \mathcal{N}$ and $\mathcal{N}_0 \subseteq |\beta|_{\mathcal{I}}$. Hence, $\mathcal{N}_0 \subseteq \mathcal{N} \to \mathcal{N}_0 \subseteq |b|_{\mathcal{I}} \to |\beta|_{\mathcal{I}} = |\alpha|_{\mathcal{I}}$.
— If $\alpha \in D^{\mathsf{exa}}$, then $b \in \mathcal{M}(\overline{D^{\mathsf{exa}}})$ and $\beta \in D^{\mathsf{exa}}$. By induction hypothesis, we have $\mathcal{N}_0 \subseteq |b|_{\mathcal{I}}$ and $|\beta|_{\mathcal{I}} \subseteq \mathcal{N}$. Hence, $|\alpha|_{\mathcal{I}} = |b|_{\mathcal{I}} \to |\beta|_{\mathcal{I}} \subseteq \mathcal{N}_0 \to \mathcal{N} \subseteq \mathcal{N}$ (this last inclusion follows from the fact that for any λ-term t, for any variable x that is not free in t, if $(t)x$ is normalisable, then t is normalisable, fact that can be proved by induction on the number of left-reductions of $(t)x$). □

Proposition 5.5.

i. Every typable λ-term in System R is head-normalisable.
ii. Let $t \in \Lambda$, $\alpha \in D^{\mathsf{exa}}$ and $\Gamma \in \Phi^{\mathsf{exa}}$ such that $\Gamma \vdash_R t : \alpha$. Then, t is normalisable.

Proof.

i. Let Γ be the context $x_1 : a_1, \ldots, x_k : a_k$. For any $\gamma \in A$, we set $\mathcal{I}(\gamma) = \mathcal{N}$, where $\mathcal{N}$ is the set of head-normalisable terms. By Lemma 5.4 (i), we have $x_1 \in |a_1|_{\mathcal{I}}, \ldots, x_k \in |a_k|_{\mathcal{I}}$. Hence, by Lemma 5.3, we have $t = t[x_1/x_1, \ldots, x_k/x_k] \in |\alpha|_{\mathcal{I}}$. Using again Lemma 5.4 (i), we obtain $|\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$.
ii. Let Γ be the context $x_1 : a_1, \ldots, x_k : a_k$. For any $\gamma \in A$, we set $\mathcal{I}(\gamma) = \mathcal{N}$, where $\mathcal{N}$ is the set of normalisable terms. By Lemma 5.4 (ii), we have $x_1 \in |a_1|_{\mathcal{I}}, \ldots, x_k \in |a_k|_{\mathcal{I}}$.

Hence, by Lemma 5.3, we have $t = t[x_1/x_1, \dots, x_k/x_k] \in |\alpha|_{\mathcal{I}}$. Using again Lemma 5.4 (ii), we obtain $|\alpha|_{\mathcal{I}} \subseteq \mathcal{N}$. □

Theorem 5.6.

i. For any $t \in \Lambda$, t is head-normalisable if and only if t is typable in System R.
ii. For any $t \in \Lambda$, t is normalisable if and only if there exist $(\Gamma, \alpha) \in \Gamma^{\mathsf{exa}} \times D^{\mathsf{exa}}$ such that $\Gamma \vdash_R t : \alpha$.

Proof.

i. Apply Proposition 5.2 (i) and Proposition 5.5 (i).
ii. Apply Proposition 5.2 (ii) and Proposition 5.5 (ii). □

This theorem is not surprising: Although System R is not considered in Dezani-Ciancaglini et al. (2005), it is quite obvious that its typing power is the same as that of the systems containing Ω considered in this paper. We can note here a difference with Systems λ and $\mathbb{I}$ already mentioned: In those systems, only strongly normalisable terms are typable. Of course, such systems characterising the strongly normalisable terms cannot be in correspondence with a denotational semantics of λ-calculus, because in a denotational semantics of λ-calculus the interpretation of a weakly normalisable term and the interpretation of its normal form are equal.

Corollary 5.7. Let v and u be two closed normal terms.

i. The λ-term $(v)u$ is head-normalisable if and only if there exist $a \in \mathcal{M}(\llbracket u \rrbracket)$ and $\alpha \in D$ such that $(a, \alpha) \in \llbracket v \rrbracket$.
ii. The λ-term $(v)u$ is normalisable if and only if there exist $a \in \mathcal{M}(\llbracket u \rrbracket)$ and $\alpha \in D^{\mathsf{exa}}$ such that $(a, \alpha) \in \llbracket v \rrbracket$.

6. Quantitative results

We now turn our attention to the quantitative aspects of reduction. The aim is to give a purely semantic account of execution time. Of course, if t' is the normal form of t, we know that $\llbracket t \rrbracket = \llbracket t' \rrbracket$, so that from $\llbracket t \rrbracket$ it is clearly impossible to determine the number of reduction steps from t to t'. Nevertheless, if v and u are two normal λ-terms, we can wonder what is the number of steps leading from $(v)u$ to its (principal head) normal form. We prove in this section that we can answer the question by only referring to $\llbracket v \rrbracket$ and $\llbracket u \rrbracket$ (Theorem 6.27).

6.1. *Type derivations for states*

We now extend the type derivations for λ-terms to type derivations for closures (Definition 6.1) and for states (Definition 6.5).

Definition 6.1. The typing rules for closures are the following ones:

$$\dfrac{}{x : [\alpha] \vdash (x, e) : \alpha}\ (*)$$

$$\dfrac{\Gamma \vdash c : \alpha}{\Gamma \vdash (x, e \cup \{x \mapsto c\}) : \alpha}\ (*)$$

$$\dfrac{\Gamma, x : a \vdash (u, e) : \alpha}{\Gamma \vdash (\lambda x.u, e) : (a, \alpha)}\ (*)$$

$$\dfrac{\Gamma_0 \vdash (v, e) : ([\alpha_1, \ldots, \alpha_n], \alpha) \qquad \Gamma_1 \vdash (u, e) : \alpha_1, \ldots, \Gamma_n \vdash (u, e) : \alpha_n}{\sum_{i=0}^{n} \Gamma_i \vdash ((v)u, e) : \alpha}$$

$(*) : x \notin \mathsf{dom}(e)$

The size $|\Pi|$ of a derivation Π is quite simply its size as a tree, i.e. the number of its nodes.

Definition 6.2. For any closure c, for any $(\Gamma, [\alpha_1, \ldots, \alpha_p]) \in \Phi \times \mathcal{M}(D)$, we define what a derivation Π of $\Gamma \vdash c : [\alpha_1, \ldots, \alpha_p]$ is and what $|\Pi|$ is for such a derivation: A derivation Π of $\Gamma \vdash c : [\alpha_1, \ldots, \alpha_p]$ is a p-tuple $(\Pi^1, \ldots, \Pi^p)$ such that there exists $(\Gamma^1, \ldots, \Gamma^p) \in \Phi^p$ and

— for $1 \leqslant i \leqslant p$, Π^i is a derivation of $\Gamma^i \vdash c : \alpha_i$;
— and $\Gamma = \sum_{i=1}^{p} \Gamma^i$.

If $\Pi = (\Pi^1, \ldots, \Pi^p)$ is a derivation of $\Gamma \vdash c : a$, then we set $|\Pi| = \sum_{i=1}^{p} |\Pi^i|$, where $|\Pi^i|$ is the size of the derivation Π^i.

Lemma 6.3. Let e be an environment. Let $x \in \mathcal{V} \setminus \mathsf{dom}(e)$. Let Π_0 be a derivation of $\Gamma_0, x : a \vdash (t, e) : \alpha$ and let Π' be a derivation of $\Gamma' \vdash c : a$. Then, there exists a derivation Π of $\Gamma_0 + \Gamma' \vdash (t, e \cup \{x \mapsto c\}) : \alpha$ such that $|\Pi| = |\Pi_0| + |\Pi'|$.

Proof. By induction on t. Notice that if $t = x$, then $|\Pi_0| = 1$ and $a = [\alpha]$, and if $t \in \mathcal{V} \setminus \{x\}$, then $a = []$ and $|\Pi'| = 0$. □

Lemma 6.4. Let e be an environment. Let $x \in \mathcal{V} \setminus \mathsf{dom}(e)$. Let Π be a derivation of $\Gamma \vdash (t, e \cup \{x \mapsto c\}) : \alpha$. There exist a derivation Π_0 of $\Gamma_0, x : a \vdash (t, e) : \alpha$ and a derivation Π' of $\Gamma' \vdash c : a$ such that $\Gamma = \Gamma_0 + \Gamma'$ and $|\Pi| = |\Pi_0| + |\Pi'|$.

Proof. By induction on t. Notice that if $t = x$, then $|\Pi_0| = 1$ and $a = [\alpha]$, and if $t \in \mathcal{V} \setminus \{x\}$, then $a = []$ and $|\Pi'| = 0$. □

Definition 6.5. Let $s = (c_0, \ldots, c_q)$ be a state. A finite sequence $(\Pi_0, \ldots, \Pi_q)$ is said to be a derivation of $\Gamma \vdash s : \alpha$ if there exist $b_1, \ldots, b_q \in \mathcal{M}(D)$, $\Gamma_0, \ldots, \Gamma_q \in \Phi$ such that

— Π_0 is a derivation of $\Gamma_0 \vdash c_0 : b_1 \ldots b_q \alpha$;
— for any $k \in \{1, \ldots, q\}$, Π_k is a derivation of $\Gamma_k \vdash c_k : b_k$;
— and $\Gamma = \sum_{k=0}^{q} \Gamma_k$.

In this case, we set $|(\Pi_0, \ldots, \Pi_q)| = \sum_{k=0}^{q} |\Pi_k|$.

6.2. *Relating size of derivations and execution time*

The aim of this subsection is to prove Theorem 6.9, that gives the exact number of steps leading to the principal head normal form by means of derivations in System R.

Proposition 6.6. Let t be a head normalisable λ-term. For any $(\Gamma,\alpha) \in \Phi \times D$, for any $\Pi \in \Delta(t,(\Gamma,\alpha))$, we have $l_h((t,\perp)\cdot\varepsilon) \leqslant |\Pi|$.

Proof. By Theorem 2.10, we can prove, by induction on $l_h(s)$, that for any $s \in \mathbb{S}$ such that $\overline{s}$ is head normalisable, for any $(\Gamma,\alpha) \in \Phi \times D$, for any derivation Π of $\Gamma \vdash s : \alpha$, we have $l_h(s) \leqslant |\Pi|$.

The base case is trivial, because we never have $l_h(s) = 0$. The inductive step is divided into five cases:

— In the case where $s = (x,e)\cdot\pi$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, $l_h(s) = 1 \leqslant |\Pi|$.

— In the case where $s = (x, e \cup \{x \mapsto c\})\cdot(c_1,\ldots,c_q)$, we have $\Pi = (\Pi_0,\ldots,\Pi_q)$, where

- Π_0 is a derivation of $\Gamma_0 \vdash (x, e\cup\{x \mapsto c\}) : b_1\ldots b_q\alpha$;
- for $1 \leqslant k \leqslant q$, Π_k is a derivation of $\Gamma_k \vdash c_k : b_k$;
- and $\Gamma = \sum_{k=0}^{q}\Gamma_k$.

There exists a derivation Π'_0 of $\Gamma_0 \vdash c : b_1\ldots b_q\alpha$ such that $|\Pi_0| = |\Pi'_0| + 1$. We have

$$\begin{aligned} l_h(s) &= l_h(c\cdot(c_1,\ldots,c_q)) + 1 \\ &\leqslant |(\Pi'_0,\Pi_1,\ldots,\Pi_q)| + 1 \\ &\quad \text{(by induction hypothesis)} \\ &= |\Pi'_0| + \sum_{k=1}^{q}|\Pi_k| + 1 \\ &= |\Pi_0| + \sum_{k=1}^{q}|\Pi_k| \\ &= |\Pi|. \end{aligned}$$

— In the case where $s = (\lambda x.u, e)\cdot(c,c_1,\ldots,c_q)$, we have $\Pi = (\Pi_0,\Pi',\Pi_1,\ldots,\Pi_q)$, where

- Π_0 is a derivation of $\Gamma_0 \vdash (\lambda x.u, e) : bb_1\ldots b_q\alpha$;
- Π' is a derivation of $\Gamma' \vdash c : b$;
- for $1 \leqslant k \leqslant q$, Π_k is a derivation of $\Gamma_k \vdash c_k : b_k$;
- $\Gamma = \sum_{k=0}^{q}\Gamma_k + \Gamma'$.

There exists a derivation Π'_0 of $\Gamma_0, x : b \vdash (u,e) : b_1\ldots b_q\alpha$ such that $|\Pi_0| = |\Pi'_0| + 1$; by Lemma 6.3, there exists a derivation Π'' of $\Gamma_0 + \Gamma' \vdash (u, e\cup\{x \mapsto c\}) : b_1\ldots b_q\alpha$ such that $|\Pi''| = |\Pi'_0| + |\Pi'|$. The sequence $(\Pi'',\Pi_1,\ldots,\Pi_q)$ is a derivation of

$$\Gamma \vdash ((u, e\cup\{x\mapsto c\}), c_1,\ldots,c_q) : \alpha.$$

We have

$$l_h(s) = l_h((u, e\cup\{x\mapsto c\})\cdot(c_1,\ldots,c_q)) + 1$$

$$\begin{aligned}
&\leqslant |(\Pi'', \Pi_1, \dots, \Pi_q)| + 1 \\
&\qquad \text{(by induction hypothesis)} \\
&= |\Pi''| + \sum_{k=1}^{q} |\Pi_k| + 1 \\
&= |\Pi'_0| + |\Pi'| + \sum_{k=1}^{q} |\Pi_k| + 1 \\
&= |\Pi_0| + |\Pi'| + \sum_{k=1}^{q} |\Pi_k| \\
&= |\Pi|.
\end{aligned}$$

— In the case where $s = ((v)u, e) \cdot (c_1, \dots, c_q)$, we have $\Pi = (\Pi_0, \dots, \Pi_q)$, where

- Π_0 is a derivation of $\Gamma_0 \vdash ((v)u, e) : b_1 \dots b_q \alpha$;
- for $1 \leqslant k \leqslant q$, Π_k is a derivation of $\Gamma_k \vdash c_k : b_k$;
- $\Gamma = \sum_{k=0}^{q} \Gamma_k$.

There exist $b \in \mathcal{M}(D)$, $\Gamma'_0, \Gamma''_0 \in \Phi$, a derivation Π'_0 of $\Gamma'_0 \vdash (v, e) : bb_1 \dots b_q \alpha$ and a derivation Π''_0 of $\Gamma''_0 \vdash (u, e) : b$ such that $\Gamma_0 = \Gamma'_0 + \Gamma''_0$ and $|\Pi_0| = |\Pi'_0| + |\Pi''_0| + 1$. The sequence $(\Pi'_0, \Pi''_0, \Pi_1, \dots, \Pi_q)$ is a derivation of $\Gamma \vdash ((v, e), (u, e), c_1, \dots, c_q) : \alpha$. We have

$$\begin{aligned}
l_h(s) &= l_h((v, e) \cdot ((u, e), c_1, \dots, c_q)) + 1 \\
&\leqslant |(\Pi'_0, \Pi''_0, \Pi_1, \dots, \Pi_q)| + 1 \\
&\qquad \text{(by induction hypothesis)} \\
&= |\Pi'_0| + |\Pi''_0| + \sum_{k=1}^{q} |\Pi_k| + 1 \\
&= |\Pi_0| + \sum_{k=1}^{q} |\Pi_k| \\
&= |(\Pi_0, \dots, \Pi_q)| \\
&= |\Pi|.
\end{aligned}$$

— In the case where $s = (\lambda x.u, e) \cdot \varepsilon$, we have $\Pi = (\Pi_0)$ with Π_0 a derivation of $\Gamma \vdash (\lambda x.u, e) : \alpha$. There exists a derivation Π'_0 of $\Gamma, x : b \vdash (u, e) : \beta$ such that $\alpha = (b, \beta)$ and $|\Pi_0| = |\Pi'_0| + 1$. The sequence (Π_0) is a derivation of $\Gamma, x : b \vdash (u, e) : \beta$. We have

$$\begin{aligned}
l_h(s) &= l_h((u, e) \cdot \varepsilon) + 1 \\
&\leqslant |\Pi'_0| + 1 \\
&\qquad \text{(by induction hypothesis)} \\
&= |\Pi|.
\end{aligned}$$

□

Proposition 6.7. Let t be a head normalisable λ-term. There exist $(\Gamma, \alpha) \in \Phi \times D$ and $\Pi \in \Delta(t, (\Gamma, \alpha))$ such that $l_h((t, \perp) \cdot \varepsilon) = |\Pi|$.

Proof. By Theorem 2.10, we can prove, by induction on $l_h(s)$, that for any $s \in \mathbb{S}$ such that $\bar{s}$ is head normalisable, there exist $(\Gamma, \alpha) \in \Phi \times \Gamma$ and a derivation Π of $\Gamma \vdash s : \alpha$ such that we have $l_h(s) = |\Pi|$.

The base case is trivial, because we never have $l_h(s) = 0$. The inductive step is divided into five cases:

— In the case where $s = (x, e) \cdot (c_1, \ldots, c_q)$ and $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, we have $l_h(s) = 1$ and there exists a derivation $\Pi = (\Pi_0, \ldots, \Pi_q)$ of $\Gamma \vdash s : \alpha$, where Π_0 is a derivation of $x : [\underbrace{[] \ldots []}_{q \text{ times}} \alpha] \vdash (x, e) : \underbrace{[] \ldots []}_{q \text{ times}} \alpha$ with $|\Pi_0| = 1$ and $|\Pi_1| = \ldots = |\Pi_q| = 0$.

— In the case where s is of the shape $(x, e) \cdot \pi$ with $x \in \mathsf{dom}(e)$, or $((v)u, e) \cdot \pi$ or $((\lambda x.u), e) \cdot \varepsilon$, apply the induction hypothesis.

— In the case where s is of the shape $(\lambda x.u, e) \cdot \pi$ with $\pi \neq \varepsilon$, apply the induction hypothesis and Lemma 6.4. □

Definition 6.8. For every $\mathfrak{D} \in \mathcal{P}(\Delta) \cup \mathcal{P}(\Delta^{<\omega})$, we set $|\mathfrak{D}| = \{|\Pi| \,/\, \Pi \in \mathfrak{D}\}$.

Theorem 6.9. For any λ-term t, we have $l_h((t, \perp) \cdot \varepsilon) = \inf |\Delta(t)|$.

Proof. We distinguish between the following two cases:

— The λ-term t is not head normalisable: By Theorem 5.6 (i), $\inf |\Delta(t)| = \infty$ and, by Theorem 2.9, $l_h((t, \perp) \cdot \varepsilon) = \infty$.

— The λ-term t is head normalisable: Apply Proposition 6.6 and Proposition 6.7. □

6.3. *Computing the normal form and* 1-*typings*

In the preceding subsection, we related $l_h(t)$ to the size of the derivations of t for any λ-term t. Now, we want to relate $l_\beta(t)$ to the size of the derivations of t. We will show that if the value of $l_\beta(t)$ is finite (i.e. t is normalisable), then $l_\beta(t)$ is the size of the least derivations of t with typings that satisfy a particular property and that, otherwise, there is no such derivation. In particular, whenever t is normalisable, $l_\beta(t)$ is the size of the derivations of t with 1-*typings*. This notion of 1-*typing*, defined in Definition 6.10, is a generalisation of the notion of *principal typing* introduced in Coppo et al. (1980).

Definition 6.10. The typing rules for deriving 1-*typings* of normal λ-terms are the following ones:

$$\frac{\Gamma, x : a \vdash_1 t : \alpha}{\Gamma \vdash_1 \lambda x.t : (a, \alpha)} \, ,$$

$$\frac{\Gamma_1 \vdash_1 u_1 : \alpha_1 \qquad \ldots \qquad \Gamma_n \vdash_1 u_n : \alpha_n}{\sum_{i=1}^{n} \Gamma_i + \{(x, [[\alpha_1] \ldots [\alpha_n]\gamma])\} \vdash_1 (x)u_1 \ldots u_n : \gamma} \, \gamma \in A \, .$$

A 1-*typing* of a normalisable λ-term is a 1-typing of its normal form.

Fact 6.11. Any normal λ-term has a 1-typing.

Proof. Any normal λ-term is of the form $\lambda x_1 \dots \lambda x_k.(x)u_1 \dots u_n$ (with $k, n \geqslant 0$), where x is a variable and $u_1, \dots, u_n$ are normal λ-terms. □

The reader acquainted with the concept of *experiment* on proof-nets in linear logic could notice that a 1-typing of a normal λ-term is the same thing as the result of what Tortora de Falco (2000) calls *a* 1-*experiment* of the proof-net obtained by the translation of this λ-term mentioned in Subsection 4.2.

Note that if t is a normalisable λ-term and (Γ, α) is a 1-typing of t, then $(\Gamma, \alpha) \in \Phi^{\mathsf{exa}} \times D^{\mathsf{exa}}$; more precisely, a typing (Γ, α) of a normalisable λ-term is a 1-typing if and only if every multiset in negative occurrence in Γ (resp. in positive occurrence in α) is a singleton.

Proposition 6.12. Let t be a normalisable λ-term. For any $(\Gamma, \alpha) \in \Phi^{\mathsf{exa}} \times D^{\mathsf{exa}}$, for any $\Pi \in \Delta(t, (\Gamma, \alpha))$, we have $l_\beta((t, \bot) \cdot \varepsilon) \leqslant |\Pi|$.

Proof. By Theorem 2.15, we can prove, by induction on $l_\beta(s)$, that for any $s \in \mathbb{S}$ such that $\overline{s}$ is normalisable, for any $(\Gamma, \alpha) \in \Phi^{\mathsf{exa}} \times D^{\mathsf{exa}}$, for any derivation Π of $\Gamma \vdash s : \alpha$, we have $l_\beta(s) \leqslant |\Pi|$.

The base case is trivial, because we never have $l_\beta(s) = 0$. The inductive step is divided into five cases:

— In the case where $s = (x, e) \cdot (c_1, \dots, c_q)$ with $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, we have $\Pi = (\Pi_0, \dots, \Pi_q)$, where Π_0 is a derivation of $\Gamma \vdash (x, e) : b_1 \dots b_q \alpha$ with $b_1, \dots, b_q \neq []$ (since $\Gamma(x) = b_1 \dots b_q \alpha \in \overline{D^{\mathsf{exa}}}$).

— The cases where $s = (x, e \cup \{x \mapsto c\}) \cdot (c_1, \dots, c_q)$ or $s = (\lambda x.u, e) \cdot (c, c_1, \dots, c_q)$ or $s = ((v)u, e) \cdot (c_1, \dots, c_q)$ or $s = (\lambda x.u, e) \cdot \varepsilon$ have to be dealt with in the same way as in the proof of Proposition 6.6. □

Proposition 6.13. Assume that t is a normalisable λ-term and that (Γ, α) is a 1-typing of t. Then, there exists a derivation Π of $\Gamma \vdash_R t : \alpha$ such that $l_\beta((t, \bot) \cdot \varepsilon) = |\Pi|$.

Proof. By Theorem 2.15, we can prove, by induction on $l_\beta(s)$, that for any $s \in \mathbb{S}$ such that $\overline{s}$ is normalisable and for any 1-typing (Γ, α) of $\overline{s}$, there exists a derivation Π of $\Gamma \vdash s : \alpha$ such that $l_\beta(s) = |\Pi|$.

The base case is trivial, because we never have $l_\beta(s) = 0$. The inductive step is divided into five cases:

— In the cases where $s = (x, e) \cdot (c_1, \dots, c_q)$ with $x \in \mathcal{V} \setminus \mathsf{dom}(e)$, (Γ, α) is a 1-typing of $(x)t_1 \dots t_q$, where $t_1, \dots, t_q$ are the respective normal forms of $\overline{c_1}, \dots, \overline{c_q}$, hence there exist $\Gamma_1, \dots, \Gamma_q$, $\alpha_1, \dots, \alpha_q$ such that

  – $\Gamma = \sum_{k=1}^{q} \Gamma_k + \{(x, [[\alpha_1] \dots [\alpha_q]\alpha])\}$

  – and $(\Gamma_1, \alpha_1), \dots, (\Gamma_q, \alpha_q)$ are 1-typings of $t_1, \dots, t_q$, respectively.

  By induction hypothesis, there exist q derivations $\Pi_1, \dots, \Pi_q$ of $\Gamma_1 \vdash c_1 : \alpha_1, \dots, \Gamma_q \vdash c_q : \alpha_q$, respectively such that $l_\beta(c_k) = |\Pi_k|$ for $1 \leqslant k \leqslant q$. We denote by Π_0 the unique derivation of $x : [[\alpha_1] \dots [\alpha_q]\alpha] \vdash (x, e) : [\alpha_1] \dots [\alpha_q]\alpha$.

Set $\Pi = (\Pi_0, \dots, \Pi_q)$: It is a derivation of $\Gamma \vdash s : \alpha$ and we have

$$\begin{aligned} l_\beta(s) &= \sum_{k=1}^{q} l_\beta(c_k) + 1 \\ &= \sum_{k=1}^{q} |\Pi_k| + 1 \\ &\quad \text{(by induction hypothesis)} \\ &= |\Pi_0| + \sum_{k=1}^{q} |\Pi_k| \\ &= |\Pi|. \end{aligned}$$

— In the cases where s is of the shape $(x, e) \cdot \pi$ with $x \in \mathsf{dom}(e)$, or $((v)u, e).\pi$, or $(\lambda x.u, e).\varepsilon$, apply the induction hypothesis.
— In the case where s is of the shape $(\lambda x.u, e) \cdot \pi$ with $\pi \neq \varepsilon$, apply the induction hypothesis and Lemma 6.4. □

Definition 6.14. For any λ-term t, we set $\Delta^{\mathsf{exa}}(t) = \bigcup_{(\Gamma,\alpha)\in\Phi^{\mathsf{exa}}\times D^{\mathsf{exa}}} \Delta(t, (\Gamma, \alpha))$.

Theorem 6.15. For any λ-term t, we have $l_\beta((t, \perp) \cdot \varepsilon) = \inf |\Delta^{\mathsf{exa}}(t)|$.

Proof. We distinguish between two cases.

— The λ-term t is not normalisable: By Theorem 5.6 (ii), $\inf |\Delta^{\mathsf{exa}}(t)| = \infty$ and, by Theorem 2.15, $l_\beta((t, \perp) \cdot \varepsilon) = \infty$.
— The λ-term t is normalisable: By Proposition 6.12, we have

$$(\forall \Pi \in \Delta^{\mathsf{exa}}(t))\; l_\beta((t, \perp) \cdot \varepsilon) \leqslant |\Pi|;$$

by Fact 6.11 and Proposition 6.13, we have $(\exists \Pi \in \Delta^{\mathsf{exa}}(t))\; l_\beta((t, \perp) \cdot \varepsilon) = |\Pi|$. □

6.4. *Relating semantics to execution time*

In this subsection, we prove the first truly semantic measure of execution time of this paper by bounding (by purely semantic means, i.e. without considering derivations) the number of steps of the computation of the principal head normal form (Theorem 6.20).

We define the size $|\alpha|$ of any $\alpha \in D$ using an auxiliary function aux.

Definition 6.16. For any $\alpha \in D$, we define $|\alpha|$ and $\mathsf{aux}(\alpha)$ by induction on $\mathsf{depth}(\alpha)$:

— if $\alpha \in A$, then $|\alpha| = 1$ and $\mathsf{aux}(\alpha) = 0$;
— if $\alpha = ([\alpha_1, \dots, \alpha_n], \alpha_0)$, then
  - $|\alpha| = \sum_{i=1}^{n} \mathsf{aux}(\alpha_i) + |\alpha_0| + 1$;
  - and $\mathsf{aux}(\alpha) = \sum_{i=1}^{n} |\alpha_i| + \mathsf{aux}(\alpha_0) + 1$.

For any $a = [\alpha_1, \dots, \alpha_n] \in \mathcal{M}(D)$, we set $|a| = \sum_{i=1}^{n} |\alpha_i|$ and $\mathsf{aux}(a) = \sum_{i=1}^{n} \mathsf{aux}(\alpha_i)$.

Notice that for any $\alpha \in D$, the size $|\alpha|$ of α is the sum of the number of positive occurrences of atoms in α and of the number of commas separating a multiset of types and a type.

Example 6.17. Let $\gamma \in A$. Set $\alpha = ([\gamma], \gamma)$ and $a = [\underbrace{\alpha, \ldots, \alpha}_{n \text{ times}}]$. We have $|(a, \alpha)| = 2n + 3$.

Lemma 6.18. For any λ-term u, if there exists a derivation Π of $x_1 : a_1, \ldots, x_m : a_m \vdash_R u : \alpha$, then $|a_1 \ldots a_m\alpha| = \mathsf{aux}(a_1 \ldots a_m\alpha)$.

Proof. By induction on Π. □

Lemma 6.19. Let v be a normal λ-term and let Π be a derivation of $x_1 : a_1, \ldots, x_m : a_m \vdash_R v : \alpha$. Then, we have $|\Pi| \leqslant |a_1 \ldots a_m\alpha|$.

Proof. By induction on v. □

Theorem 6.20. Let v and u be two closed normal λ-terms. Assume $(a, \alpha) \in [\![v]\!]$ and $\mathsf{Supp}(a) \subseteq [\![u]\!]$.

i. We have $l_h(((v)u, \bot) \cdot \varepsilon) \leqslant 2|a| + |\alpha| + 2$.
ii. If, moreover, $\alpha \in D^{\mathsf{exa}}$, then we have

$$l_\beta(((v)u, \bot) \cdot \varepsilon) \leqslant 2|a| + |\alpha| + 2.$$

Proof. Set $a = [\alpha_1, \ldots, \alpha_n]$. There exist a derivation Π_0 of $\vdash_R v : (a, \alpha)$ and n derivations $\Pi_1, \ldots, \Pi_n$ of $\vdash_R u : \alpha_1, \ \ldots, \vdash_R u : \alpha_n$, respectively. Hence, there exists a derivation Π of $\vdash_R (v)u : \alpha$ such that $|\Pi| = \sum_{i=0}^{n} |\Pi_i| + 1$.

i. We have

$$\begin{aligned} l_h(((v)u, \bot) \cdot \varepsilon) &\leqslant \sum_{i=0}^{n} |\Pi_i| + 1 \\ &\quad \text{(by Proposition 6.6)} \\ &\leqslant |(a, \alpha)| + \sum_{i=1}^{n} |\alpha_i| + 1 \\ &\quad \text{(by Lemma 6.19)} \\ &= \sum_{i=1}^{n} \mathsf{aux}(\alpha_i) + |\alpha| + 1 + |a| + 1 \\ &= \sum_{i=1}^{n} |\alpha_i| + |\alpha| + 1 + |a| + 1 \\ &\quad \text{(by Lemma 6.18)} \\ &= 2|a| + |\alpha| + 2. \end{aligned}$$

ii. The only difference with the proof of (i) is that we apply Proposition 6.12 instead of Proposition 6.6. □

6.5. *The exact number of steps*

This subsection is devoted to giving the exact number of steps of computation by purely semantic means. For arbitrary points $(a, \alpha) \in [\![v]\!]$ such that $a \in \mathcal{M}([\![u]\!])$, it is clearly impossible to obtain an equality in Theorem 6.20, because there exist such points with different sizes.

The only equalities we have by now are Theorems 6.9 6.15, which use the size of the derivations. A first idea is then to look for points $(a, \alpha) \in [\![v]\!]$ such that $a \in \mathcal{M}([\![u]\!])$ with $|(a, \alpha)|$ equals to the sizes of the derivations used in these theorems. But there are cases in which such points do not exist.

A more subtle way out is nevertheless possible, and here is where the notions of equivalence between derivations and of substitution defined in Subsection 4.3 come into the picture. More precisely, using the notion of substitution, Proposition 6.24 (the only place where we use the non-finiteness of the set A of atoms through Fact 6.21 and Lemma 6.23) shows how to find, for any $\beta \in [\![t]\!]$, an element $\alpha \in [\![t]\!]$ such that $|\alpha| = \min |\Delta(t, \beta)|$.

We remind that $A = D \setminus (\mathcal{M}(D) \times D)$. The equivalence relation $\sim$ has been defined in Definition 4.8 and the notion of substitution has been defined in Definition 4.9. We recall that we denote by $\mathcal{S}$ the set of substitutions.

Fact 6.21. Let v be a normal λ-term and let Π be a derivation of

$$x_1 : b_1, \ldots, x_m : b_m \vdash_R v : \beta.$$

There exist $a_1, \ldots, a_m, \alpha$ and a derivation Π' of $x_1 : a_1, \ldots, x_m : a_m \vdash_R v : \alpha$ such that $\Pi' \sim \Pi$ and $|\Pi'| + m = |a_1 \ldots a_m \alpha|$. If, moreover, A is infinite, then we can choose Π' in such a way that there exists a substitution σ such that $\overline{\sigma}(a_1) = b_1, \ldots, \overline{\sigma}(a_m) = b_m$ and $\sigma(\alpha) = \beta$.

Proof. By induction on v. □

In the case where A is infinite, the derivation Π' of the lemma is what (Coppo et al. 1980) calls a *ground deduction for* v.

Definition 6.22. For every $X \in \mathcal{P}(D) \cup \mathcal{P}(\mathcal{M}(D))$, we set $|X| = \{|\alpha| \ / \ \alpha \in X\}$.

Lemma 6.23. Assume A is infinite. Let t be a closed normal λ-term, let $\beta \in D$ and let $\Pi \in \Delta(t, \beta)$. Then, we have

$$|\Pi| = \min |\{\alpha \in D \ / \ (\exists \sigma \in \mathcal{S})\ \sigma(\alpha) = \beta \text{ and } (\exists \Pi' \in \Delta(t, \alpha))\ \Pi' \sim \Pi\}|.$$

Proof. Apply Lemma 6.19 and Fact 6.21. □

Proposition 6.24. Assume A is infinite. Let t be a closed normal λ-term and let $\beta \in [\![t]\!]$. We have $\min |\Delta(t, \beta)| = \min |\{\alpha \in [\![t]\!] \ / \ (\exists \sigma \in \mathcal{S})\sigma(\alpha) = \beta\}|$.

Proof. Set $m = \min|\Delta(t,\beta)|$ and $n = \min|\{\alpha \in [\![t]\!] \,/\, (\exists\sigma \in \mathcal{S})\sigma(\alpha) = \beta\}|$.

First, we prove that $m \leqslant n$. Let $\alpha \in [\![t]\!]$ such that we have $(\exists\sigma \in \mathcal{S})\sigma(\alpha) = \beta$. By Theorem 4.2, $\Delta(t,\alpha) \neq \emptyset$: let $\Pi' \in \Delta(t,\alpha)$. By Proposition 4.10, there exists $\Pi \in \Delta(t,\beta)$ such that $\Pi \sim \Pi'$. By Lemma 6.19, we have $|\Pi'| \leqslant |\alpha|$. Hence, we obtain $m \leqslant |\Pi| = |\Pi'| \leqslant |\alpha|$.

Now, we prove the inequality $n \leqslant m$. Let $\Pi \in \Delta(t,\beta)$.

$$\begin{aligned} n &= \min|\{\alpha \in D \,/\, (\exists\Pi' \in \Delta(t,\alpha))(\exists\sigma \in \mathcal{S})\sigma(\alpha) = \beta\}| \\ &\quad \text{(by Theorem 4.2)} \\ &\leqslant \min|\{\alpha \in D \,/\, (\exists\Pi' \in \Delta(t,\alpha))\,(\exists\sigma \in \mathcal{S})\,(\Pi' \sim \Pi \text{ and } \sigma(\alpha) = \beta)\}| \\ &= |\Pi| \\ &\quad \text{(by Lemma 6.23).} \end{aligned}$$

□

Corollary 6.25. Assume A is infinite. Let t be a closed normal λ-term and let $b \in \mathcal{M}([\![t]\!])$. We have $\min|\Delta(t,b)| = \min|\{a \in \mathcal{M}([\![t]\!]) \,/\, (\exists\sigma \in \mathcal{S})\,\overline{\sigma}(a) = b\}|$.

The point of Theorem 6.27 is that the number of steps of the computation of the (principal head) normal form of $(v)u$, where v and u are two closed normal λ-terms, can be determined from $[\![v]\!]$ and $[\![u]\!]$.

Definition 6.26. For any $X, Y \subseteq D$, we denote by $\mathcal{U}(X,Y)$ the set

$$\{((a,\alpha),a') \in (X \setminus A) \times \mathcal{M}(Y) \,/\, (\exists\sigma \in \mathcal{S})\,\overline{\sigma}(a) = \overline{\sigma}(a')\}$$

and by $\mathcal{U}^{\mathsf{exa}}(X,Y)$ the set

$$\left\{((a,\alpha),a') \in (X \setminus A) \times \mathcal{M}(Y) \,/\, (\exists\sigma \in \mathcal{S})(\overline{\sigma}(a) = \overline{\sigma}(a') \text{ and } \sigma(\alpha) \in D^{\mathsf{exa}})\right\}.$$

Theorem 6.27. Assume A is infinite. For any two closed normal λ-terms u and v, we have

i. $l_h(((v)u,\perp).\varepsilon) = \inf\{|(a,\alpha)| + |a'| + 1 \,/\, ((a,\alpha),a') \in \mathcal{U}([\![v]\!],[\![u]\!])\}$;
ii. $l_\beta(((v)u,\perp).\varepsilon) = \inf\{|(a,\alpha)| + |a'| + 1 \,/\, ((a,\alpha),a') \in \mathcal{U}^{\mathsf{exa}}([\![v]\!],[\![u]\!])\}$.

Proof.

i. We distinguish between two cases.

— If $\Delta((v)u) = \emptyset$, then Theorem 6.9 shows that $l_h(((v)u,\perp).\varepsilon) = \infty$ and Theorem 4.2 and Proposition 4.10 show that $\mathcal{U}([\![v]\!],[\![u]\!]) = \emptyset$.

— Else, we have

$$\begin{aligned} & l_h(((v)u,\perp)\cdot\varepsilon) \\ &= \min\{|\Pi| + |\Pi'| + 1 \,/\, (\Pi,\Pi') \in \bigcup_{(b,\beta)\in\mathcal{M}(D)\times D} (\Delta(v,(b,\beta)) \times \Delta(u,b))\} \\ &\quad \text{(by Theorem 6.9)} \\ &= \min\{|(a,\alpha)| + |a'| + 1 \,/\, ((a,\alpha),a') \in \mathcal{U}([\![v]\!],[\![u]\!])\} \\ &\quad \text{(by applying Proposition 6.24 and Corollary 6.25, and by noticing} \\ &\quad \text{that the atoms in } a \text{ can be assumed distinct from those in } a'). \end{aligned}$$

ii. We distinguish between two cases.

— If $\Delta^{\mathsf{exa}}((v)u) = \emptyset$, then Theorem 6.15 shows that $l_\beta(((v)u, \perp) \cdot \varepsilon) = \infty$ and Theorem 4.2 and Proposition 4.10 show that $\mathcal{U}^{\mathsf{exa}}([\![v]\!], [\![u]\!]) = \emptyset$.

— Otherwise, we have

$$\begin{aligned} & l_\beta(((v)u, \perp) \cdot \varepsilon) \\ = \; & \min\{|\Pi| + |\Pi'| + 1 \;/\; (\Pi, \Pi') \in \bigcup_{(b,\beta) \in \mathcal{M}(D) \times D^{\mathsf{exa}}} (\Delta(v, (b, \beta)) \times \Delta(u, b))\} \\ & \text{(by Theorem 6.15)} \\ = \; & \min\{|(a, \alpha)| + |a'| + 1 \;/\; ((a, \alpha), a') \in \mathcal{U}^{\mathsf{exa}}([\![v]\!], [\![u]\!])\} \\ & \text{(by applying Proposition 6.24 and Corollary 6.25, and by noticing} \\ & \text{that the atoms in } a \text{ can be assumed distinct from those in } a'). \end{aligned}$$

□

Example 6.28. Set $v = \lambda x.(x)x$ and $u = \lambda y.y$. Let $\gamma_0, \gamma_1 \in A$. Set

— $\alpha = \gamma_0$;

— $a = [\gamma_0, ([\gamma_0], \gamma_0)]$;

— $a' = [([\gamma_1], \gamma_1), ([\gamma_2], \gamma_2)]$.

Let σ be a substitution such that $\sigma(\gamma_0) = ([\gamma_0], \gamma_0)$, $\sigma(\gamma_1) = \gamma_0$ and $\sigma(\gamma_2) = ([\gamma_0], \gamma_0)$. We have

— $(a, \alpha) \in [\![v]\!]$;

— $\mathsf{Supp}(a') \subseteq [\![u]\!]$;

— $\overline{\sigma}(a) = \overline{\sigma}(a')$;

— $|(a, \alpha)| = 4$ and $|a'| = 4$.

By Example 2.7, we know that we have $l_h(((v)u, \perp) \cdot \varepsilon) = 9$. And we have $|(a, \alpha)| + |a'| + 1 = 9$.

The following example shows that the assumption that A is infinite is necessary.

Example 6.29. Let n be a non-zero integer. Set $I = \lambda y.y$ and $v = \lambda x.(x)\underbrace{I \ldots I}_{n \text{ times}}$. We have $\mathcal{U}([\![v]\!], [\![I]\!]) \subseteq \{(([[([\alpha_1], \alpha_1)] \ldots [([\alpha_n], \alpha_n)]\alpha], \alpha), [([\alpha_0], \alpha_0)]) \;/\; \alpha_0, \ldots, \alpha_n, \alpha \in D\}$. Let $\gamma_0, \ldots, \gamma_n, \delta \in A$ distinct. We have

$$(([[([\gamma_1], \gamma_1)] \ldots [([\gamma_n], \gamma_n)]\delta], \delta), [([\gamma_0], \gamma_0)]) \in \mathcal{U}^{\mathsf{exa}}([\![v]\!], [\![I]\!]).$$

Indeed, just consider any $\sigma \in \mathcal{S}$ defined by setting $\sigma(\gamma_i) = \alpha_{n-i}$ and $\sigma(\delta) = \alpha_1$ in such a way that $\alpha_{i+1} = ([\alpha_i], \alpha_i)$.

Hence, for any $\alpha_0, \ldots, \alpha_n, \alpha \in D$, if there exists $i \in \{0, \ldots, n\}$ such that $\alpha_i \notin A$, then we have

$$\begin{aligned} |([[([\alpha_1], \alpha_1)] \ldots [([\alpha_n], \alpha_n)]\alpha], \alpha)| + |[([\alpha_0], \alpha_0)]| + 1 &> l_\beta(((v)u, \perp) \cdot \varepsilon) \\ &= l_h(((v)u, \perp) \cdot \varepsilon). \end{aligned}$$

On the other hand, if $\gamma_0, \ldots, \gamma_n \in A$, $\alpha \in D$ and there exist $i, j \in \{0, \ldots, n\}$ such that $i \neq j$ and $\gamma_i = \gamma_j$, then we have

$$(([[([\gamma_1], \gamma_1)] \ldots [([\gamma_n], \gamma_n)]\alpha], \alpha), [([\gamma_0], \gamma_0)]) \notin \mathcal{U}([\![v]\!], [\![I]\!]).$$

All this shows that if $\mathsf{Card}(A) = n$, then we do not have

$$l_h(((v)I, \perp) \cdot \varepsilon) = \inf\{|(a, \alpha)| + |a'| + 1 \ / \ ((a, \alpha), a') \in \mathcal{U}([\![v]\!], [\![I]\!])\};$$

nor $l_\beta(((v)I, \perp) \cdot \varepsilon) = \inf\{|(a, \alpha)| + |a'| + 1 \ / \ ((a, \alpha), a') \in \mathcal{U}^{\mathsf{exa}}([\![v]\!], [\![I]\!])\}$.

Note that, as the following example illustrates, the non-idempotency is crucial.

Example 6.30. For any integer $n \geqslant 1$, set $\overline{n} = \lambda f.\lambda x.\underbrace{(f)\dots(f)}_{n \text{ times}} x$ and $I = \lambda y.y$. Let $\gamma \in A$. Set $\alpha = ([\gamma], \gamma)$ and $a = [\underbrace{\alpha, \dots, \alpha}_{n \text{ times}}]$. We have $(a, \alpha) \in [\![\overline{n}]\!]$ and $\alpha \in [\![I]\!]$. We have $l_h(((\overline{n})I, \perp) \cdot \varepsilon) = 4(n+1) = 2n + 3 + 2n + 1 = |(a, \alpha)| + |a| + 1$ (see Example 6.17). But with *idempotent* types (as in System $\mathcal{D}$), for any integers $p, q \geqslant 1$, we would have $\mathcal{U}(\overline{p}, I) = \mathcal{U}(\overline{q}, I)$ (any Church integer $\overline{n}$, for $n \geqslant 1$ has type $((\gamma \to \gamma) \to (\gamma \to \gamma))$ in System $\mathcal{D}$).

This work is partially the result of discussions with Thomas Ehrhard, I warmly thank him; I also thank Patrick Baillot and Kazushige Terui too for stimulating discussions. Lastly, I am very grateful to the anonymous referees for their careful reading of the paper.

References

Baillot, P. and Terui, K. (2004). Light types for polynomial time computation in lambda-calculus. In: *Proceedings of LICS 2004*, IEEE Computer Society Press, 266–275.

Barendregt, H.P. (1984). *The Lambda Calculus. Its Syntax and Semantics*, revised edition, North-Holland.

Benton, P.N., Bierman, G.M., de Paiva, V.C.V. and Hyland, J.M.E. (1992). Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge.

Bierman, G.M. (1993). On intuitionistic linear logic. PhD thesis, University of Cambridge.

Bierman, G.M. (1995). What is a categorical model of intuitionistic linear logic? In: *Proceedings of Conference on Typed Lambda Calculi and Applications*, vol. 902, Springer-Verlag.

Boudol, G., Curien, P.-L. and Lavatelli, C. (1999). A semantics for lambda calculi with resources. *Mathematical Structures in Comuter Science* **9** (4) 437–482.

Bucciarelli, A. and Ehrhard, T. (2001). On phase semantics and denotational semantics: The exponentials. *Annals of Pure and Applied Logic* **109** (3) 205–241.

Bucciarelli, A., Ehrhard, T. and Manzonetto, G. (2007). Not enough points is enough. In: Duparc, J. and Henzinger, T.A. (eds.) CSL'07: Proceedings of 16th Computer Science Logic. *Springer Lecture Notes in Computer Science* **4646** 268–282.

de Carvalho, D. (2006). Execution time of lambda-terms via non-uniform semantics and intersection types. Preprint, Institut de Mathématiques de Luminy.

de Carvalho, D. (2007). Sémantiques de la logique linéaire et temps de calcul. PhD thesis, Université Aix-Marseille 2.

de Carvalho, D., Pagani, M. and Tortora de Falco, L. (2008). A semantic measure of the execution time in linear logic. *Theoretical Computer Science* **412** (20) 1884–1902.

Coppo, M., Dezani-Ciancaglini, M. and Venneri, B. (1980). Principal type schemes and λ-calculus semantics. In: Seldin, J.P. and Hindley, J.R. (eds.) *To H. B. Curry : Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, 535–560.

Danos, V. and Regnier, L. (2004). Head linear reduction. Manuscript.

Dezani-Ciancaglini, M., Honsell, F. and Motohama, Y. (2005). Compositional characterisations of λ-terms using intersection types. *Theoretical Computer Science* **340** (3) 459–496.

Ehrhard, T. and Regnier, L. (2006a). Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science* **364** (2) 166–195.

Ehrhard, T. and Regnier, L. (2006b). Böhm trees, Krivine's machine and the Taylor expansion of lambda-terms. In: Beckmann, A., Berger, U., Löwe, B. and Tucker, J.V. (eds.) *Logical Approaches to Computational Barriers, Proceedings of the 2nd Conference on Computability in Europe,* CiE 2006, Swansea, UK, June 30–July 5, 2006, Springer-Verlag, 186–197.

Girard, J.Y. (1986). The system F of variable types, fifteen years later. *Theoretical Computer Science* **45** (2) 159–192.

Girard, J.Y. (1987). Linear logic. *Theoretical Computer Science* **50** (1) 1–102.

Guerrini, S. (2004). Proof nets and the λ-calculus. In: Ehrhard, T., Girard, J.-Y., Ruet, P. and Scott, P. (eds.) *Linear Logic in Computer Science*, Cambridge University Press, 65–118.

Kfoury, A.J. (2000). A linearization of the Lambda-calculus and consequences. *Journal of Logic and Computation* **10** (3) 411–436.

Kfoury, A.J., Mairson, H.G., Turbak, F. A. and Wells, J.B. (1999). Relating typability and expressiveness in finite-rank intersection types systems (Extended Abstract). In: *ICFP*, 90–101.

Krivine, J.L. (1990). *Lambda-Calcul Types et Modèles*, Masson.

Krivine, J.L. (2007). A call-by-name lambda-calculus machine. *Higher Order and Symbolic Computation* **20** (3) 199–207.

Lafont, Y. (1988). Logiques, catégories et machines. PhD thesis, Université Paris 7.

Lambek, J. and Scott, P.J. (1986). *Introduction to Higher Order Categorical Logic*, Cambridge University Press.

Mac Lane, S. (1998). *Categories for the Working Mathematician*, Springer-Verlag.

Mascari, G.F. and Pedicini, M. (1994). Head linear reduction and pure proof net extraction. *Theoretical Computer Science* **135** (1) 111–137.

Neergaard, P.M. and Mairson, H.G. (2004). Types, potency, and idempotency: Why nonlinearity and amnesia make a type system work. In: *ICFP '04: Proceedings of the 9th ACM SIGPLAN International Conference on Functional Programming*, ACM Press, 138–149.

Regnier, L. (1992). Lambda-calcul et réseaux. PhD thesis, Université Paris 7.

Ronchi Della Rocca, S. (1988). Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science* **59** 181–209.

Seely, R. (1989). Linear logic, $*$-autonomous categories and cofree coalgebras. In: *Categories in Computer Science and Logic*, Contemporary Mathematics 92, American Mathematical Society, 371–382.

Selinger, P. (2002). The lambda calculus is algebraic. *Journal of Functional Programming* **12** (6) 549–566.

Tortora de Falco, L. (2000). Réseaux, cohérence et expériences obsessionnelles. PhD thesis, Université Paris 7.