IO vs OI in Higher-Order Recursion Schemes

Axel Haddad
LIAFA (Université Paris 7 & CNRS) & LIGM (Université Paris Est & CNRS)

We propose a study of the modes of derivation of higher-order recursion schemes, proving that value trees obtained from schemes using innermost-outermost derivations (IO) are the same as those obtained using unrestricted derivations.

Given that higher-order recursion schemes can be used as a model of functional programs, innermost-outermost derivations policy represents a theoretical view point of call by value evaluation strategy.

1 Introduction

Recursion schemes have been first considered as a model of computation, representing the syntactical aspect of a recursive program [15, 2, 3, 4] . At first, (order-1) schemes were modelling simple recursive programs whose functions only take values as input (and not functions). Since, higher-order versions of recursion schemes [11, 5, 6, 7, 8, 9] have been studied.

More recently, recursion schemes were studied as generators of infinite ranked trees and the focus was on deciding logical properties of those trees [12, 8, 10, 1, 13, 14].

As for programming languages, the question of the evaluation policy has been widely studied. Indeed, different policies results in the different evaluation [8, 9, 7]. There are two main evaluations policy for schemes: outermost-innermost derivations (OI) and inner-outermost IO derivations, respectively corresponding to call by need and call by value in programming languages.

Standardization theorem for the lambda-calculus shows that for any scheme, outermost-innermost derivations (OI) lead to the same tree as unrestricted derivation. However, this is not the case for IO derivations. In this paper we prove that the situation is different for schemes. Indeed, we establish that the trees produced using schemes with IO policy are the same as those produced using schemes with OI policy. For a given a scheme of order n, we can use a simplified continuation passing style transformation, to get a new scheme of order $n+1$ in which IO derivations will be the same as OI derivations in the initial scheme (Section 3). Conversely, in order to turn a scheme into another one in which unrestricted derivations lead to the same tree as IO derivations in the initial scheme, we adapt Kobayashi's [13] recent results on HORS model-checking, to compute some key properties over terms (Section 4.1). Then we embed these properties into a scheme turning it into a self-correcting scheme of the same order of the initial scheme, in which OI and IO derivations produce the same tree (Section 4.2).

2 Preliminaries

Types are defined by the grammar $\tau ::= o \mid \tau \to \tau$; o is called the ***ground type***. Considering that $\to$ is associative to the right (i.e. $\tau_1 \to (\tau_2 \to \tau_3)$ can be written $\tau_1 \to \tau_2 \to \tau_3$), any type τ can be written uniquely as $\tau_1 \to ... \to \tau_k \to o$. The integer k is called the ***arity*** of τ. We define the ***order of a type*** by $\text{order}(o) = 0$ and $\text{order}(\tau_1 \to \tau_2) = \max(\text{order}(\tau_1)+1, \text{order}(\tau_2))$. For instance $o \to o \to o \to o$ is a type

D. Miller and Z. Ésik (Eds.): Fixed Points
in Computer Science 2012 (FICS 2012)
EPTCS 77, 2012, pp. 23–30, doi:10.4204/EPTCS.77.4

of order 1 and arity 3, $(o \to o) \to (o \to o)$, that can also be written $(o \to o) \to o \to o$ is a type of order 2. Let $\tau^\ell \to \tau'$ be a shortcut for $\underbrace{\tau \to ... \to \tau}_{\ell \text{ times}} \to \tau'$.

Let Γ be a finite set of symbols such that to each symbol is associated a type. Let Γ^τ denote the set of symbols of type τ. For all type τ, we define the set of ***terms*** of type $\mathscr{T}^\tau(\Gamma)$ as the smallest set satisfying: $\Gamma^\tau \subseteq \mathscr{T}^\tau(\Gamma)$ and $\bigcup_{\tau'}\{t\ s \mid t \in \mathscr{T}^{\tau' \to \tau}(\Gamma), s \in \mathscr{T}^{\tau'}(\Gamma)\} \subseteq \mathscr{T}^\tau(\Gamma)$. If a term t is in $\mathscr{T}^\tau(\Gamma)$, we say that t has type τ. We shall write $\mathscr{T}(\Gamma)$ as the set of terms of any type, and $t : \tau$ if t has type τ. The arity of a term t, $arity(t)$, is the arity of its type. Remark that any term t can be uniquely written as $t = \alpha\ t_1 ... t_k$ with $\alpha \in \Gamma$. We say that α is the ***head*** of the term t. For instance, let $\Gamma = \{F : (o \to o) \to o \to o\ ,\ G : o \to o \to o\ ,\ H : (o \to o)\ ,\ \mathtt{a} : o\}$: $F\ H$ and $G\ \mathtt{a}$ are terms of type $o \to o$; $F(G\ \mathtt{a})\ (H\ (H\ a))$ is a term of type o; $F\ \mathtt{a}$ is not a term since F is expecting a first argument of type $o \to o$ while $\mathtt{a}$ has type o.

Let $t : \tau$, $t' : \tau'$ be two terms, $x : \tau'$ be a symbol of type τ', then we write $t_{[x \mapsto t']} : \tau$ the term obtained by substituting all occurences of x by t' in the term t. A ***τ-context*** is a term $C[\bullet^\tau] \in \mathscr{T}(\Gamma \uplus \{\bullet^\tau : \tau\})$ containing exactly one occurrence of $\bullet^\tau$; it can be seen as an application turning a term into another, such that for all $t : \tau$, $C[t] = C[\bullet^\tau]_{[\bullet^\tau \mapsto t]}$. In general we will only talk about ground type context where $\tau = o$ and we will omit to specify the type when it is clear. For instance, if $C[\bullet] = F \bullet\ (H\ (H\ a))$ and $t' = G\ \mathtt{a}$ then $C[t'] = F\ (G\ \mathtt{a})\ (H\ (H\ \mathtt{a}))$.

Let Σ be a set of symbols of order at most 1 (i.e. each symbols has type o or $o \to ... \to o$) and $\bot : o$ be a fresh symbol. A ***tree*** t over $\Sigma \uplus \bot$ is a mapping $t : dom^t \to \Sigma \uplus \bot$, where dom^t is a prefix-closed subset of $\{1,...,m\}^*$ such that if $u \in dom^t$ and $t(u) = a$ then $\{j \mid uj \in dom^t\} = \{1,...,arity(a)\}$. Note that there is a direct bijection between ground terms of $\mathscr{T}^o(\Sigma \uplus \bot)$ and finite trees . Hence we will freely allow ourselves to treat ground terms over $\Sigma \uplus \bot$ as trees. We define the partial order $\sqsubseteq$ over trees as the smallest relation satisfying $\bot \sqsubseteq t$ and $t \sqsubseteq t$ for any tree t, and $a\ t_1 ... t_k \sqsubseteq a\ t'_1 ... t'_k$ iff $t_i \sqsubseteq t'_i$. Given a (possibly infinite) sequence of trees $t_0, t_1, t_2, ...$ such that $t_i \sqsubseteq t_{i+1}$ for all i, one can prove that the set of all t_i has a supremum that is called the ***limit tree*** of the sequence.

A ***higher order recursion scheme (HORS)*** $G = \langle \mathscr{V}, \Sigma, \mathscr{N}, \mathscr{R}, S \rangle$ is a tuple such that: $\mathscr{V}$ is a finite set of typed symbols called ***variables***; Σ is a finite set of typed symbols of order at most 1, called the ***set of terminals***; $\mathscr{N}$ is a finite set of typed symbols called ***set of non-terminals***; $\mathscr{R}$ is a set of ***rewrite rules***, one per non terminal $F : \tau_1 \to ... \to \tau_k \to o \in \mathscr{N}$, of the form $F\ x_1\ ...\ x_k\ \to\ e$ with $e : o \in \mathscr{T}(\Sigma \uplus \mathscr{N} \uplus \{x_1, ..., x_k\})$; $S \in \mathscr{N}$ is the ***initial non-terminal***.

We define the ***rewriting relation*** $\to_G \in\ \mathscr{T}(\Sigma \uplus \mathscr{N})^2$ (or just $\to$ when G is clear) as $t \to_G t'$ iff there exists a context $C[\bullet]$, a rewrite rule $F\ x_1 ... x_k \to e$, and a term $F\ t_1\ ...\ t_k : o$ such that $t = C[F\ t_1 ... t_k]$ and $t' = C[e_{[x_1 \mapsto t_1] ... [x_k \mapsto t_k]}]$. We call $F\ t_1\ ...\ t_k : o$ a ***redex***. Finally we define $\to_G^*$ as the reflexive and transitive closure of $\to_G$.

We define inductively the ***$\bot$-transformation*** $(\cdot)^\bot\ :\ \mathscr{T}^o(\mathscr{N} \uplus \Sigma) \to \mathscr{T}^o(\Sigma \uplus \{\bot : o\})$: $(F\ t_1\ ...\ t_k)^\bot = \bot\ \forall F \in \mathscr{N}$ and $(a\ t_1\ ...\ t_k)^\bot\ =\ a\ t_1^\bot ... t_k^\bot$ for all $a \in \Sigma$. We define a ***derivation***, as a possibly infinite sequence of terms linked by the rewrite relation. Let $t_0 = S \to_G t_1 \to_G t_2 \to_G ...$ be a derivation, then one can check that $(t_0)^\bot \sqsubseteq (t_1)^\bot \sqsubseteq (t_2)^\bot \sqsubseteq ...$, hence it admits a limit. One can prove that the set of all such limit trees has a greatest element that we denote $\|G\|$ and refer to as the ***value tree*** of G. Note that $\|G\|$ is the supremum of $\{t^\bot \mid S \to^* t\}$. Given a term $t : o$, we denote by G_t the scheme obtained by transforming G such that it starts derivations with the term t, formally, $G_t = \langle \mathscr{V}, \Sigma, \mathscr{N} \uplus \{S'\}, \mathscr{R} \uplus \{S' \to t\}, S' \rangle$. One can prove that if $t \to t'$ then $\|G_t\| = \|G_{t'}\|$.

Example. Let $G = \langle \mathscr{V}, \Sigma, \mathscr{N}, \mathscr{R}, S \rangle$ be the scheme such that: $\mathscr{V} = \{x : o, \phi : o \to o, \psi : (o \to o) \to o \to o\}$, $\Sigma = \{\mathtt{a} : o^3 \to o, \mathtt{b} : o \to o \to o, \mathtt{c} : o\}$, $\mathscr{N} = \{F : \big((o \to o) \to o \to o\big) \to (o \to o) \to o \to o, H : (o \to$

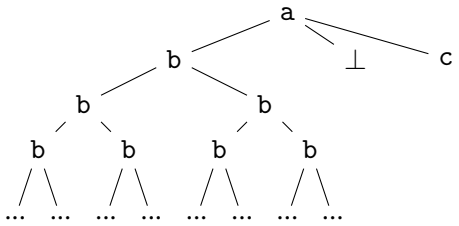
$o) \to o \to o, I, J, K : o \to o, S : o\}$, and $\mathscr{R}$ contains the following rewrite rules:

$$\begin{array}{llll} F\ \psi\ \phi\ x & \to & \psi\ \phi\ x & I\ x \to x \quad K\ x \to K\ (K\ x) \\ J\ x & \to & \mathtt{b}\ (J\ x)\ (J\ x) & H\ \phi\ x \to \mathtt{a}\ (J\ x)\ (K\ x)\ (\phi\ x) \\ & & & S \to F\ H\ I\ \mathtt{c} \end{array}$$

Here is an example of finite derivation:

$$\begin{array}{lllllll} S & \to & F\ H\ I\ \mathtt{c} & \to & H\ I\ \mathtt{c} & \to & \mathtt{a}\ (J\ \mathtt{c})\ (K\ \mathtt{c})\ (I\ \mathtt{c}) \\ & & & & & \to & \mathtt{a}\ (J\ \mathtt{c})\ (K\ (K\ \mathtt{c}))\ (I\ \mathtt{c}) \quad \to \quad \mathtt{a}\ (J\ \mathtt{c})\ (K\ (K\ (K\ \mathtt{c})))\ (I\ \mathtt{c}) \end{array}$$

If one extends it by always rewriting a redex of head K, its limit is the tree $\mathtt{a}\ \bot\ \bot\ \bot$, but this is not the value tree of G. The value tree $\|G\|$ is depicted below.



Evaluation Policies

We now put constraints on the derivations we allow. If there are no constraints, then we say that the derivations are unrestricted and we let $\mathrm{Acc}^G = \{t : o \mid S \to^* t\}$ be the set of accessible terms using unrestricted derivations. Given a rewriting $t \to t'$ such that $t = C[F\ s_1\ ...\ s_k]$ and $t' = C[e_{[\forall j\ x_j \mapsto s_j]}]$ with $F\ x_1 ... x_k \to e \in \mathscr{R}$.

- We say that $t \to t'$ is an ***outermost-innermost*** (OI) rewriting (written $t \to_{OI} t'$) there is no redex containing the occurrence of $\bullet$ as a subterm of $C[\bullet]$.
- We say that $t \to t'$ is an ***innermost-outermost*** (IO) rewriting (written $t \to_{IO} t'$), if for all j there is no redex as a subterm of s_j.

Let $\mathrm{Acc}^G_{OI} = \{t : o \mid S \to^*_{OI} t\}$ be the set of accessible terms using OI derivations and $\mathrm{Acc}^G_{IO} = \{t : o \mid S \to^*_{IO} t\}$ be the set of accessible terms using IO derivations. There exists a supremum of Acc^G_{OI} (resp. Acc^G_{IO}) which is the maximum of the limit trees of OI derivations(resp. IO derivations). We write it $\|G\|_{OI}$ (resp. $\|G\|_{IO}$). For all recursive scheme G, $(\mathrm{Acc}^G)^\bot = (\mathrm{Acc}^G_{OI})^\bot$, in particular $\|G\|_{OI} = \|G\|$. But $\|G\|_{IO} \sqsubseteq \|G\|$ and in general, the equality does not hold (see the example is the next section).

3 From OI to IO

Fix a recursion scheme $G = \langle \mathscr{V}, \Sigma, \mathscr{N}, \mathscr{R}, S \rangle$. Our goal is to define another scheme $\overline{G} = \langle \overline{\mathscr{V}}, \Sigma, \overline{\mathscr{N}}, \overline{\mathscr{R}}, I \rangle$ such that $\|\overline{G}\|_{IO} = \|G\|$. The idea is to add an extra argument (Δ) to each non terminal, that will be required to rewrite it (hence the types are changed). We feed this argument to the outermost non terminal, and duplicate it to subterms only if the head of the term is a terminal. Hence all derivations will be IO-derivations.

We define the $\overline{(\cdot)}$ ***transformation*** over types by $\overline{o} = o \to o$, and $\overline{\tau_1 \to \tau_2} = \overline{\tau_1} \to \overline{\tau_2}$. In particular, if $\tau = \tau_1 \to ... \to \tau_k \to o$ then $\overline{\tau} = \overline{\tau_1} \to ... \to \overline{\tau_k} \to o \to o$. Note that for all τ, $\mathrm{order}(\overline{\tau}) = \mathrm{order}(\tau) + 1$.

For all $x:\tau\in\mathscr{V}$ we define $\overline{x}:\overline{\tau}$ as a fresh variable. Let ar_{max} be the maximum arity of terminals, we define $\eta_1,...,\eta_{arity_{max}}:o\to o$ and $\delta:o$ as fresh variables, and we let $\overline{\mathscr{V}}=\{\overline{x}:\overline{\tau}\mid x\in\mathscr{V}\}\uplus\{\eta_1,...,\eta_{ar_{max}}\}\uplus\{\delta:o\}$. Note that δ is the only variable of type o. For all $a:\tau\in\Sigma$ define $\overline{a}:\overline{\tau}$ as a fresh **non-terminal** and for all $F:\tau\in\mathscr{N}$ define $\overline{F}:\overline{\tau}$ as a fresh non-terminal. Let $\overline{\mathscr{N}}=\{\overline{a}:\overline{\tau}\mid a\in\Sigma\}\uplus\{\overline{F}:\overline{\tau}\mid F\in\mathscr{N}\}\uplus\{\Delta:o,I:o\}$. Note that I and Δ are the only symbols in $\overline{\mathscr{N}}$ of type o.

Let $t:\tau\in\mathscr{T}(\mathscr{V}\uplus\Sigma\uplus\mathscr{N})$, we define inductively the term $\overline{t}:\overline{\tau}\in\mathscr{T}(\overline{\mathscr{V}}\uplus\overline{\mathscr{N}})$: If $t=x\in\mathscr{V}$ (resp. $t=a\in\Sigma$, $t=F\in\mathscr{N}$), we let $\overline{t}=\overline{x}\in\overline{\mathscr{V}}$ (resp. $\overline{t}=\overline{a}\in\overline{\Sigma}$, $\overline{t}=\overline{F}\in\mathscr{N}$), if $t=t_1\ t_2:\tau$ then $\overline{t}=\overline{t_1}\ \overline{t_2}$.

Let $F\ x_1\ ...\ x_k\to e$ be a rewrite rule of $\mathscr{R}$. We define the (valid) rule $\overline{F}\ \overline{x_1}\ ...\ \overline{x_k}\ \delta\ \to\ \overline{e}\ \Delta$ in $\overline{\mathscr{R}}$. Let $a\in\Sigma$ of arity k, we define the rule $\overline{a}\ \eta_1\ ...\ \eta_k\ \delta\ \to\ a\ (\eta_1\ \Delta)\ ...\ (\eta_k\ \Delta)$ in $\overline{\mathscr{R}}$. We also add the rule $I\ \to\ \overline{S}\ \Delta$ to $\overline{\mathscr{R}}$. Finally let $\overline{G}=\langle\overline{\mathscr{V}},\Sigma,\overline{\mathscr{N}},\overline{\mathscr{R}},I\rangle$.

Example. Let $G=\langle\mathscr{V},\Sigma,\mathscr{N},\mathscr{R},S\rangle$ be the order-1 recursion scheme with $\Sigma=\{\mathtt{a},\mathtt{c}:o\}$, $\mathscr{N}=\{S:o,F:o\to o\to o,H:o\to o\}$, $\mathscr{V}=\{x,y:o\}$, and the following rewrite rules:

$$S\ \to\ F\ (H\ \mathtt{a})\ \mathtt{c}\qquad F\ x\ y\ \to\ y\qquad H\ x\ \to\ H\ (H\ x)$$

Then we have $\|G\|_{OI}=\mathtt{c}$ while $\|G\|_{IO}=\bot$ (indeed, the only IO derivation is the following $S\to F\ (Ha)\ \mathtt{c}\to F\ (H\ (H\ a))\ \mathtt{c}\to F\ (H\ (H\ (H\ a)))\ \mathtt{c}\to ...$). The order-2 recursion scheme $\overline{G}=\langle\overline{\mathscr{V}},\Sigma,\overline{\mathscr{N}},\overline{\mathscr{R}},I\rangle$ is given by $\overline{\mathscr{N}}=\{I,\Delta:o,\overline{S},\overline{\mathtt{a}},\overline{\mathtt{c}}:o\to o,\overline{F}:(o\to o)\to(o\to o)\to o\to o,\overline{H}:(o\to o)\to o\to o\}$,$\overline{\mathscr{V}}=\{\delta:o,\overline{x},\overline{y}:o\to o\}$ and the following rewrite rules:

$$\begin{array}{llllllll} I & \to & \overline{S}\ \Delta & \overline{S}\ \delta & \to & \overline{F}\ (\overline{H}\ \overline{a})\ \overline{c}\ \Delta & \overline{F}\ \overline{x}\ \overline{y}\ \delta & \to & \overline{y}\ \Delta \\ \overline{H}\ \overline{x}\ \delta & \to & \overline{H}\ (\overline{H}\ \overline{x})\ \Delta & \overline{\mathtt{c}}\ \delta & \to & \mathtt{c} & \overline{\mathtt{a}}\ \delta & \to & \mathtt{a} \end{array}$$

Note that in the term $\overline{F}\ (\overline{H}\ \overline{\mathtt{a}})\ \overline{\mathtt{c}}\ \Delta$, the subterm $\overline{H}\ \overline{\mathtt{a}}$ is no longer a redex since it lacks its last argument, hence it cannot be rewritten, then the only IO derivation, which is the only unrestricted derivation is $I\to\overline{S}\ \Delta\to\overline{F}\ (\overline{H}\ \overline{\mathtt{a}})\ \overline{\mathtt{c}}\ \Delta\to\overline{\mathtt{c}}\ \Delta\to\mathtt{c}$. Therefore $\|\bar{G}\|_{IO}=\|\bar{G}\|=\mathtt{c}=\|G\|$.

Lemma 1. *Any derivation of $\overline{G}$ is in fact an OI **and** an IO derivation. Hence that $\|\overline{G}\|_{IO}=\|\overline{G}\|$.*

Proof (Sketch). The main idea is that the only redexes will be those that have Δ as last argument of the head non-terminal. The scheme is constructed so that Δ remains only on the outermost non-terminals, that is why any derivation is an OI derivation. Furthermore, we have that if $t=\overline{F}\ t_1...t_k\Delta$ is a redex, then none of the t_i contains Δ, therefore they do not contain any redex, hence t is an innermost redex. □

Note that *OI* derivations in $\overline{G}$ acts like *OI* derivations in G, hence $\|G\|=\|\overline{G}\|$.

Theorem 2 (OI vs IO)**.** *Let G be an order-n scheme. Then one can construct an order-$(n+1)$ scheme $\overline{G}$ such that $\|G\|=\|\overline{G}\|_{IO}$.*

4 From IO to OI

The goal of this section is to transform the scheme G into a scheme G'' such that $\|G''\|=\|G\|_{IO}$. The main difference between IO and OI derivations is that some redex would lead to $\bot$ in IO derivation while OI derivations could be more productive. For example take $F:o\to o$ such that $F\ x\to c$, and $H:o$ such that $H\to a\ H$, with $a:o\to o$ and $c:o$ being terminal symbols. The term $F\ H$ has a unique *OI* derivation, $F\ H\to_{OI} c$, it is finite and it leads to the value tree assiocated. On the other hand, the (unique) IO derivation is the following $F\ H\to F(a\ H)\to F\ (a\ (a\ H))\to ...$ which leads to the tree $\bot$.

The idea of the transformation is to compute a tool (based on a type system) that decides if a redex would produce $\bot$ with *IO* derivations (Section 4.1); then we embed it into G and force any such redex to produce $\bot$ even with unrestricted derivations (Section 4.2).

4.1 The Type System

Given a term $t : \tau \in \mathscr{T}(\Sigma \uplus \mathscr{N})$, we define the two following properties on t: $\mathscr{P}_\perp(t)$ ="The term t has type o and its associated IO valuation tree is $\perp$", and $\mathscr{P}_\infty(t)$ ="the term t has not necessarily ground type, it contains a redex r such that any IO derivation from r producing it's IO valuation tree is infinite". Note that $\mathscr{P}_\infty(t)$ is equivalent to "the term t contains a redex r such that $\|G_r\|_{IO}$ is either infinite or contains $\perp$". In this section we describe a type system, inspired from the work of Kobayashi [13], that characterises if a term verifies these properties.

Let Q be the set $\{q_\perp, q_\infty\}$. Given a type τ, we define inductively the sets $(\tau)^{atom}$ and $(\tau)^\wedge$ called respectively set of atomic mappings and set of conjunctive mappings:
$(o)^{atom} = Q$, $(o)^\wedge = \{\bigwedge\{\theta_1,...,\theta_i\} \mid \theta_1,...,\theta_i \in Q\}$, $(\tau_1 \to \tau_2)^{atom} = \{q_\infty\} \uplus \{(\tau_1)^\wedge \to (\tau_2)^{atom}\}$
$(\tau_1 \to \tau_2)^\wedge = \{\bigwedge\{\theta_1,...,\theta_i\} \mid \theta_1,...,\theta_i \in (\tau_1 \to \tau_2)^{atom}\}$.

We will usually use the letter θ to represents atomic mappings, and the letter σ to represent conjunctive mappings. Given a conjunctive mapping σ (resp. an atomic mapping θ) and a type τ, we write $\sigma :: \tau$ (resp. $\theta ::_a \tau$) the relation $\sigma \in (\tau)^\wedge$ (resp. $\theta \in (\tau)^{atom}$). For the sake of simplicity, we identify the atomic mapping θ with the conjunctive mapping $\bigwedge\{\theta\}$.

Given a term t and a conjunctive mapping σ, we define a judgment as a tuple $\Theta \vdash t \triangleright \sigma$, pronounce "from the environment Θ, one can prove that t matches the conjunctive mapping σ", where the environment Θ is a partial mapping from $\mathscr{V} \uplus \mathscr{N}$ to conjunctive mapping. Given an environment Θ, $\alpha \in \mathscr{V} \uplus \mathscr{N}$ and a conjunctive mapping σ, we define the environment $\Theta' = \Theta, \alpha \triangleright \sigma$ as $Dom(\Theta') = Dom(\Theta) \cup \{\alpha\}$ and $\Theta'(\alpha) = \sigma$ if $\alpha \notin Dom(\Theta)$, $\Theta'(\alpha) = \sigma \wedge \Theta(\alpha)$ otherwise, and $\Theta'(\beta) = \Theta(\beta)$ if $\beta \neq \alpha$.

We define the following judgement rules:

$$\frac{\Theta \vdash t \triangleright \theta_1 \quad ... \quad \Theta \vdash t \triangleright \theta_n}{\Theta \vdash t \triangleright \bigwedge\{\theta_1,...,\theta_n\}}(Set) \qquad \frac{}{\Theta, \alpha \triangleright \bigwedge\{\theta_1,...,\theta_n\} \vdash \alpha \triangleright \theta_i}(At) \text{ (for all } i\text{)}$$

$$\frac{}{\Theta \vdash a \triangleright \sigma_1 \to ... \to \sigma_{i \leq arity(a)} \to q_\infty}(\Sigma) \text{ (for } a \in \Sigma \text{ and } \exists j\ \sigma_j = q_\infty \text{)}$$

$$\frac{\Theta \vdash t_1 \triangleright \sigma \to \theta \quad \Theta \vdash t_2 \triangleright \sigma}{\Theta \vdash t_1\ t_2 \triangleright \theta}(App) \quad \frac{}{\Theta \vdash t \triangleright q_\infty \to q_\infty}(q_\infty \to q_\infty\ I) \text{ (if } t : \tau_1 \to \tau_2\text{)} \quad \frac{\Theta \vdash t_1 \triangleright q_\infty}{\Theta \vdash t_1\ t_2 \triangleright q_\infty}(q_\infty\ I)$$

Remark that there is no rules that directly involves $q_\perp$, but it does not mean that no term matches $q_\perp$, since it can appear in Θ. Rules like (At) or (App) may be used to state that a term matches $q_\perp$.

We say that (G,t) matches the conjunctive mapping σ written $\vdash (G,t) \triangleright \sigma$ if there exists an environment Θ, called a witness environment of $\vdash (G,t) \triangleright \sigma$, such that (1) $Dom(\Theta) = \mathscr{N}$, (2) $\forall F : \tau \in \mathscr{N}\ \Theta(F) :: \tau$, (3) if $F\ x_1...x_k \to e \in \mathscr{R}$ and $\Theta \vdash F \triangleright \sigma_1 \to ... \to \sigma_{i \leq k} \to q$ then either there exists j such that $q_\infty \in \sigma_j$, or $i = k$ and $\Theta, x_1 \triangleright \sigma_1, ..., x_k \triangleright \sigma_k \vdash e \triangleright q$, (4) $\Theta \vdash t \triangleright \sigma$.

The following two results state that this type system matches the properties $\mathscr{P}_\perp$ and $\mathscr{P}_\infty$ and furthermore we can construct a universal environment, $\Theta^\star$, that can correctly judge any term.

Theorem 3 (Soundness and Completeness)**.** *Let G be an HORS, and t be term (of any type), $\vdash (G,t) \triangleright q_\infty$ (resp. $\vdash (G,t) \triangleright q_\perp$) if and only if $\mathscr{P}_\infty(t)$ (resp. $\mathscr{P}_\perp(t)$) holds.*

Proposition 4 (Universal Witness)**.** *There exists an environment $\Theta^\star$ such that for all term t, the judgment $\vdash (G,t) \triangleright \sigma$ holds if and only if $\Theta^\star \vdash t \triangleright \sigma$.*

Proof (Sketch). To compute $\Theta^\star$, we start with an environment Θ_0 satisfying Properties (1) and (2) ($Dom(\Theta_0) = \mathscr{N}$ and $\forall F : \tau \in \mathscr{N}\ \Theta_0(F) :: \tau$) that is able to judge any term $t : \tau$ with any conjunctive mapping $\sigma :: \tau$.

Then let $\mathscr{F}$ be the mapping from the set of environments to itself, such that for all $F : \tau_1 \to ... \to \tau_k \to o \in \mathscr{N}$, if $F\ x_1 ... x_k \to e \in \mathscr{R}$ then,

$$\begin{aligned}\mathscr{F}(\Theta)(F) = \{\sigma_1 \to ... \to \sigma_k \to q \mid q \in Q \wedge \forall i\ \sigma_i :: \tau_i \wedge \Theta, x_1 \triangleright \sigma_1, ..., x_k \triangleright \sigma_k \vdash e : q\} \\ \cup \{\sigma_1 \to ... \to \sigma_{i\leq k} \to q_\infty \mid \wedge \forall i\ \sigma_i :: \tau_i \wedge \exists j\ q_\infty \in \sigma_j\} \\ \cup \{\sigma_1 \to ... \to \sigma_k \to q_\bot \mid \forall i\ \sigma_i :: \tau_i \wedge \exists j\ q_\infty \in \sigma_j\}.\end{aligned}$$

We iterate $\mathscr{F}$ until we reach a fixpoint. The environment we get is $\Theta^\star$, it verifies properties (1) (2) and (3). Furthermore we can show that this is the maximum of all environment satisfying these properties, i.e. if $\vdash (G,t) \triangleright \sigma$ then $\Theta^\star \vdash t \triangleright \sigma$. □

4.2 Self-Correcting Scheme

For all term $t : \tau \in \mathscr{T}(\Sigma \uplus \mathscr{N})$, we define $[\![t]\!] \in (\tau)^\wedge$, called the semantics of t, as the conjunction of all atomic mappings θ such that $\Theta^\star \vdash t \triangleright \theta$ (recall that $\Theta^\star$ is the environment of Proposition 4). In particular $\mathscr{P}_\bot(t)$ (resp. $\mathscr{P}_\infty(t)$) holds if and only if $q_\bot \in [\![t]\!]$ (resp. $q_\infty \in [\![t]\!]$). Given two terms $t_1 : \tau_2 \to \tau$ and $t_2 : \tau_2$ the only rules we can apply to judge $\Theta^\star \vdash t_1\ t_2 \triangleright \theta$ are (App), $(q_\infty \to q_\infty\ I)$ and $(q_\infty\ I)$. We see that θ only depends on which atomic mappings are matched by t_1 and t_2. In other words $[\![t_1\ t_2]\!]$ only depends on $[\![t_1]\!]$ and $[\![t_2]\!]$, we write $[\![t_1]\!]\ [\![t_2]\!] = [\![t_1\ t_2]\!]$.

In this section, given a scheme $G = \langle \mathscr{V}, \Sigma, \mathscr{N}, \mathscr{R}, S\rangle$, we transform it into $G' = \langle \mathscr{V}', \Sigma, \mathscr{N}', \mathscr{R}', S\rangle$ which is basically the same scheme except that while it is producing an IO derivation, it evaluates $[\![t']\!]$ for any subterm t' of the current term and label t' with $[\![t']\!]$. Note that if $t \to_{IO} t'$, then $[\![t]\!] = [\![t']\!]$. Since we cannot syntactically label terms, we will label all symbols by the semantics of their arguments, *e.g.* if we want to label $F\ t_1 ... t_k$, we will label F with the k-tuple $([\![t_1]\!], ..., [\![t_k]\!])$.

A problem may appear if some of the arguments are not fully applied, for example imagine we want to label $F\ H$ with $H : o \to o$. We will label F with $[\![H]\!]$, but since H has no argument we do not know how to label it. The problem is that we cannot wait to label it because once a non-terminal is created, the derivation does not deal explicitly with it. The solution is to create one copy of H per possible semantics for its argument (here there are four of them: $\bigwedge\{\}, \bigwedge\{q_\bot\}, \bigwedge\{q_\infty\}, \bigwedge\{q_\bot, q_\infty\}$). This means that $F^{[\![H]\!]}$ would not have the same type as F: F has type $(o \to o) \to o$, but $F^{[\![G]\!]}$ will have type $(o \to o)^4 \to o$. Hence, $F\ H$ will be labelled the following way: $F^{[\![H]\!]}\ H^{\wedge\{\}} H^{\wedge\{q_\bot\}} H^{\wedge\{q_\infty\}} H^{\wedge\{q_\bot, q_\infty\}}$. Note that even if F has 4 arguments, it only has to be labelled with one semantics since all four arguments represent different labelling of the same term. We now formalize these notions.

Let us generalize the notion of semantics to deals with terms containing some variables. Given an environment on the variables $\Theta^{\mathscr{V}}$ such that $Dom(\Theta^{\mathscr{V}}) \subseteq \mathscr{V}$ and if $x : \tau$ then $\Theta^{\mathscr{V}}(x) :: \tau$, and given a term $t : \tau \in \mathscr{T}(\Sigma \uplus \mathscr{N} \uplus Dom(\Theta^{\mathscr{V}}))$, we define $[\![t]\!]_{\Theta^{\mathscr{V}}} \in (\tau)^\wedge$, as the conjunction of all atomic mappings θ such that $\Theta^\star, \Theta^{\mathscr{V}} \vdash t \triangleright \theta$. Given two terms $t_1 : \tau_2 \to \tau$ and $t_2 : \tau_2$ we still have that $[\![t_1\ t_2]\!]_{\Theta^{\mathscr{V}}}$ only depends on $[\![t_1]\!]_{\Theta^{\mathscr{V}}}$ and $[\![t_2]\!]_{\Theta^{\mathscr{V}}}$.

To a type $\tau = \tau_1 \to ... \to \tau_k \to o$ we associate the integer $\lceil\tau\rceil = Card(\{(\sigma_1, ..., \sigma_k) \mid \forall i\ \sigma_i \in (\tau_i)^\wedge\})$ and a complete ordering of $\{(\sigma_1, ..., \sigma_k) \mid \forall i\ \sigma_i \in (\tau_i)^\wedge\}$ denoted $\vec{\sigma}_1^\tau$, $\vec{\sigma}_2^\tau$, ... , $\vec{\sigma}_{\lceil\tau\rceil}^\tau$. We define inductively the type $\tau^+ = (\tau_1^+)^{\lceil\tau_1\rceil} \to ... \to (\tau_k^+)^{\lceil\tau_k\rceil} \to o$.

To a non terminal $F : \tau_1 \to ... \to \tau_k \to o$ (resp. a variable $x : \tau_1 \to ... \to \tau_k \to o$) and a tuple $\sigma_1 :: \tau_1, ..., \sigma_k :: \tau_k$, we associate the non-terminal $F^{\sigma_1,...,\sigma_k} : \tau_1^{\lceil \tau_1 \rceil} \to ... \to \tau_k^{\lceil \tau_k \rceil} \to o \in \mathscr{N}'$ (resp. a variable $x^{\sigma_1,...,\sigma_k} : \tau_1^{\lceil \tau_1 \rceil} \to ... \to \tau_k^{\lceil \tau_k \rceil} \to o \in \mathscr{V}'$).

Given a term $t : \tau = \tau_1 \to ... \to \tau_k \to o \in \mathscr{T}(\mathscr{V} \uplus \Sigma \uplus \mathscr{N})$ and an environment on the variables $\Theta^{\mathscr{V}}$ such that $Dom(\Theta^{\mathscr{V}}) \subseteq \mathscr{V}$ contains all variables in t, we define inductively the term $t_{\Theta^{\mathscr{V}}}^{+\sigma_1,...,\sigma_k} : \tau^+ \in \mathscr{T}(\mathscr{V}' \uplus \Sigma' \uplus \mathscr{N}')$ for all $\sigma_1 :: \tau_1, ..., \sigma_k :: \tau_k$. If $t = F \in \mathscr{N}$ (resp. $t = x \in \mathscr{V}$), $t_{\Theta^{\mathscr{V}}}^{+\sigma_1,...,\sigma_k} = F^{\sigma_1,...,\sigma_k}$ (resp. $t_{\Theta^{\mathscr{V}}}^{+\sigma_1,...,\sigma_k} = x^{\sigma_1,...,\sigma_k}$), if $t = a \in \Sigma$, $t_{\Theta^{\mathscr{V}}}^{+\sigma_1,...,\sigma_k} = a$. Finally consider the case where $t = t_1\ t_2$ with $t_1 : \tau' \to \tau$ and $t_2 : \tau'$. Let $\sigma = [\![t_2]\!]_{\Theta^{\mathscr{V}}}$. Remark that ${t_1}_{\Theta^{\mathscr{V}}}^{+\sigma,\sigma_1,...,\sigma_k} : (\tau'^+)^{\lceil \tau' \rceil} \to \tau^+$. We define $(t_1\ t_2)_{\Theta^{\mathscr{V}}}^{+\sigma_1,...,\sigma_k} = {t_1}_{\Theta^{\mathscr{V}}}^{+\sigma,\sigma_1,...,\sigma_k}\ {t_2}_{\Theta^{\mathscr{V}}}^{+\vec{\sigma}_1^{\tau'}}\ ...\ {t_2}_{\Theta^{\mathscr{V}}}^{+\vec{\sigma}_{\lceil \tau' \rceil}^{\tau'}}$. Note that since this transformation is only duplicating and anotating, given a term $t^{+\sigma_1,...,\sigma_k}$ we can uniquely find the unique term t associated to it.

Let $F : \tau_1 \to ... \to \tau_k \to o \in \mathscr{N}$, $\sigma_1 :: \tau_1, ..., \sigma_k :: \tau_k$, and $\Theta^{\mathscr{V}} = x_1 \triangleright \sigma_1, ..., x_k \triangleright \sigma_k$. If $F\ x_1 ... x_k \to e \in \mathscr{R}$, we define in $\mathscr{R}'$ the rule $F^{\sigma_1,...,\sigma_k}\ x_1^{+\vec{\sigma}_1^{\tau_1}}\ ...\ x_1^{+\vec{\sigma}_{\lceil \tau_1 \rceil}^{\tau_1}}\ ...\ x_k^{+\vec{\sigma}_1^{\tau_k}}\ ...\ x_k^{+\vec{\sigma}_{\lceil \tau_k \rceil}^{\tau_k}} \to e_{\Theta^{\mathscr{V}}}^+$. Finally, recall that $G' = \langle \mathscr{V}', \Sigma, \mathscr{N}', \mathscr{R}', S \rangle$.

The following theorem states that G' is just a labeling version of G and that it acts the same.

Theorem 5 (Equivalence between G and G')**.** *Given a term $t : o$, $\|G'_{t^+}\|_{IO} = \|G_t\|_{IO}$.*

We transform G' into the scheme G'' that will directly turn into $\perp$ a redex t such that $q_\perp \in [\![t]\!]$. For technical reason, instead of adding $\perp$ we add a non terminal $Void : o$ and a rule $Void \to Void$. $G' = \langle \mathscr{V}', \Sigma, \mathscr{N}' \uplus \{Void : o\}, \mathscr{R}'', S \rangle$ such that $\mathscr{R}''$ contains the rule $Void \to Void$ and for all $F \in \mathscr{N}$, if $q_\perp \in [\![F]\!]\ \sigma_1\ ...\ \sigma_k$ then $F^{\sigma_1,...,\sigma_k}\ x_1^{+\vec{\sigma}_1^{\tau_1}}\ ...\ x_1^{+\vec{\sigma}_{\lceil \tau_1 \rceil}^{\tau_1}}\ ...\ x_k^{+\vec{\sigma}_1^{\tau_k}}\ ...\ x_k^{+\vec{\sigma}_{\lceil \tau_k \rceil}^{\tau_k}} \to Void$ otherwise we keep the rule of $\mathscr{R}'$.

The following theorem concludes Section 4.

Theorem 6 (IO vs OI)**.** *Let G be a higher-order recursion scheme. Then one can construct a scheme G'' having the same order of G such that $\|G''\| = \|G\|_{IO}$.*

Proof (Sketch). First, given a term $t : o$, one can prove that $\|G''_{t^+}\|_{IO} = \|G'_{t^+}\|_{IO}$.

Then take a redex t such that $\|G''_t\|_{IO} = \perp$, i.e. $q_\perp \in [\![G_t]\!]$. There is only one OI derivation from t: $t \to Void \to Void \to ...$, then $\|G''_t\| = \perp$. We can extend this result saying that if there is the symbol $\perp$ at node u in $\|G''_t\|_{IO}$, then there is $\perp$ at node u in $\|G''_t\|$. Hence, since $\|G''_t\|_{IO} \sqsubseteq \|G''_t\|$, we have $\|G''\| = \|G''\|_{IO}$. Then $\|G''\| = \|G''\|_{IO} = \|G'\|_{IO} = \|G\|_{IO}$.

□

5 Conclusion

We have shown that value trees obtained from schemes using innermost-outermost derivations (IO) are the same as those obtained using unrestricted derivations. More precisely, given an order-n scheme G we create an order-$(n+1)$ scheme $\overline{G}$ such that $\|\overline{G}\|_{IO} = \|G\|$. However, the increase of the order seems unavoidable. We also create an order-n scheme G'' such that $\|\overline{G}''\| = \|G\|_{IO}$. In this case the order does not increase, however the size of the scheme deeply increases while it remains almost the same in $\overline{G}$.

References

[1] Klaus Aehlig (2006): *A Finite Semantics of Simply-Typed Lambda Terms for Infinite Runs of Automata*. In: *"Proc. of Computer Science Logic, 20th Annual Conference of the EACSL"*, *Lecture Notes in Comput. Sci.* 4207, Springer-Verlag, pp. 104–118, doi:10.1007/11874683_7.

[2] Bruno Courcelle (1978): *A Representation of Trees by Languages I*. *Theoret. Comput. Sci.* 6, pp. 255–279, doi:10.1016/0304-3975(78)90008-7.

[3] Bruno Courcelle (1978): *A Representation of Trees by Languages II*. *Theoret. Comput. Sci.* 7, pp. 25–55, doi:10.1016/0304-3975(78)90039-7.

[4] Bruno Courcelle & Maurice Nivat (1978): *The Algebraic Semantics of Recursive Program Schemes*. In: *Proc. 7th Symposium, Mathematical Foundations of Computer Science 1978*, *Lecture Notes in Comput. Sci.* 64, Springer-Verlag, pp. 16–30.

[5] Werner Damm (1977): *Higher type program schemes and their tree languages*. In: *Theoretical Computer Science, 3rd GI-Conference*, *Lecture Notes in Comput. Sci.* 48, Springer-Verlag, pp. 51–72.

[6] Werner Damm (1977): *Languages Defined by Higher Type Program Schemes*. In: *Proc. 4th Colloq. on Automata, Languages, and Programming (ICALP)*, *Lecture Notes in Comput. Sci.* 52, Springer-Verlag, pp. 164–179.

[7] Werner Damm (1982): *The IO- and OI-Hierarchies*. *Theoret. Comput. Sci.* 20, pp. 95–207, doi:10.1016/0304-3975(82)90009-3.

[8] Joost Engelfriet & Erik Meineche Schmidt (1977): *IO and OI. I*. *J. Comput. System Sci.* 15(3), pp. 328–353, doi:10.1016/S0022-0000(77)80034-2.

[9] Joost Engelfriet & Erik Meineche Schmidt (1978): *IO and OI. II*. *J. Comput. System Sci.* 16(1), pp. 67–99, doi:10.1016/0022-0000(78)90051-X.

[10] Matthew Hague, Andrzej S. Murawski, C.-H. Luke Ong & Olivier Serre (2008): *Collapsible Pushdown Automata and Recursion Schemes*. In: *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society, pp. 452–461.

[11] Klaus Indermark (1976): *Schemes with Recursion on Higher Types*. In: *Proc. 5th Symposium, Mathematical Foundations of Computer Science 1976*, *Lecture Notes in Comput. Sci.* 45, Springer-Verlag, pp. 352–358.

[12] Teodor Knapik, Damian Niwiński & Pawel Urzyczyn (2002): *Higher-Order Pushdown Trees Are Easy*. In: *Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, *Lecture Notes in Comput. Sci.* 2303, Springer-Verlag, pp. 205–222, doi:10.1007/3-540-45931-6_15.

[13] Naoki Kobayashi (2009): *Types and higher-order recursion schemes for verification of higher-order programs*. In: *Proc. 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, ACM, pp. 416–428.

[14] Naoki Kobayashi & C.-H. Luke Ong (2009): *A Type System Equivalent to the Modal Mu-Calculus Model Checking of Higher-Order Recursion Schemes*. In: *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society, pp. 179–188.

[15] M. Nivat (1972): *On the interpretation of recursive program schemes*. In: *Symposia Matematica*.