(HO)RPO Revisited

Frédéric Blanqui

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

(HO)RPO Revisited

Frédéric Blanqui

N° 5972

30 August 2006

Thème SYM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5972--FR+ENG

(HO)RPO Revisited

Frédéric Blanqui

Thème SYM — Systèmes symboliques
Projet Protheo

Rapport de recherche n° 5972 — 30 August 2006 — 20 pages

Abstract: The notion of computability closure has been introduced for proving the termination of the combination of higher-order rewriting and beta-reduction. It is also used for strengthening the higher-order recursive path ordering. In the present paper, we study in more details the relations between the computability closure and the (higher-order) recursive path ordering. We show that the first-order recursive path ordering is equal to an ordering naturally defined from the computability closure. In the higher-order case, we get an ordering containing the higher-order recursive path ordering whose well-foundedness relies on the correctness of the computability closure. This provides a simple way to extend the higher-order recursive path ordering to richer type systems.

Key-words: termination, ordering, lambda-calculus, rewriting

(HO)RPO Revisité

Résumé : La notion de clôture de calculabilité a été introduite pour prouver la terminaison de la combinaison de récriture d'ordre supérieur et de beta-réduction. Elle est aussi utilisée pour enrichir l'ordre récursif sur les chemins (RPO) à l'ordre supérieur (HORPO). Dans cet article, nous étudions la relation entre la clôture de calculabilité et (HO)RPO. Nous montrons que RPO est égal à un ordre naturellement définit à partir de la clôture. A l'ordre supérieur, nous obtenons un ordre contenant HORPO dont la preuve de bonne fondation repose sur la correction de la clôture. Cela fournit une manière simple d'étendre HORPO à des systèmes de types plus riches.

Mots-clés : terminaison, ordre, lambda-calcul, réécriture

1 Introduction

We are interested in automatically proving the termination of the combination of β-reduction and higher-order rewrite rules. There are two important approaches to higher-order rewriting: rewriting on $\beta\eta$-equivalence classes (or $\beta\overline{\eta}$-normal forms) [22] with higher-order pattern-matching (higher-order unification on higher-order patterns has been proved decidable in [23]), and the combination of β-reduction and term rewriting with higher-order pattern-matching [18]. The relation between both has been studied in [27]. The second approach is more atomic since a rewrite step in the first approach can be directly encoded by a rewrite step together with β-steps in the second approach. In this paper, we consider the second approach, restricted to first-order pattern-matching (we do not have abstractions in rule left-hand side).

The combination of β-reduction and rewriting is naturally used in dependent type systems and proof assistants implementing the proposition-as-type and proof-as-object paradigm. In these systems, two propositions equivalent modulo β-reduction and rewriting are considered as equivalent (*e.g.* $P(2+2)$ and $P(4)$). This is essential for enabling users to formalize large proofs with many computations, as recently shown by Gonthier and Werner's proof of the Four Color Theorem in the Coq proof assistant. However, for the system to be able to check the correctness of user proofs, it must at least be able to check the equivalence of two terms. Hence, the necessity to have termination criteria for the combination of β-reduction with a set R of higher-order rewrite rules.

To our knowledge, the first termination criterion for such a combination is Jouannaud and Okada's General Schema [12, 13]. It is based on Tait's technique for proving the strong normalization of the simply-typed λ-calculus [25]. Roughly speaking, since proving the strong β-normalization of simply-typed λ-terms by induction on the term structure does not work directly, Tait's idea was to prove a stronger property that he called *strong computability.* Extending Tait's technique to higher-order rewriting consists in proving that function symbols are computable too, that is, that every function call is computable whenever its arguments so are. This naturally leads to the following question: which operations preserve computability? From a set of such operations, one can define the *computability closure* of a term t, written $\mathrm{CC}_R(t)$, as the set of terms that are computable whenever t so is. Then, to get normalization, it suffices to check that, for every rule $f\vec{l} \to r$, r belongs to the computability closure of $\vec{l}$. The General Schema was implicitly doing this. The first definition of computability closure appeared in an 1997 unpublished note of Jouannaud and Okada which served as a basis for [8], an extension to dependent types of the computability closure. The computability closure was later extended to higher-order pattern-matching [5], type-level rewriting [2, 7] and rewriting modulo AC [4]. Examples of computability-preserving operations are:

- application: if $u \in \mathrm{CC}_R(t)$ and $v \in \mathrm{CC}_R(t)$, then $uv \in \mathrm{CC}_R(t)$).
- abstraction: if $u \in \mathrm{CC}_R(t)$, then $\lambda xu \in \mathrm{CC}_R(t)$).
- recursive calls on structurally smaller arguments: if $\vec{u} \in \mathrm{CC}_R(f\vec{t})$ and $\vec{u} \lhd \vec{t}$, then $f\vec{u} \in \mathrm{CC}_R(f\vec{t})$.

– reduction: if $u \in \mathrm{CC}_R(t)$ and $u \rightarrow_R v$, then $v \in \mathrm{CC}_R(t)$.

Another way to prove the termination of a set of rules is to find a decidable well-founded rewrite relation containing these rules. A well known such relation in the first-order case is the (inductively defined) recursive path ordering [24, 11] whose well-foundedness proof was initially based on Kruskal theorem [19]. The first attempts [20, 21, 15] made for generalizing this ordering to the higher-order case were not able to orient Gödel system T for instance. Finally, in 1999, Jouannaud and Rubio succeeded in defining such an ordering [14] by following the termination proof technique developed in [13]. By the way, this provided the first well-foundedness proof of RPO not based on Kruskal theorem. HORPO has also been extended to dependent types later in [28].

Although the computability closure on one hand, and the recursive path ordering on the other hand, shares the same computability-based techniques, there has been no precise comparison between these two termination criteria. In [29], one can find examples of rules that are accepted by one criterion but not the other. And Jouannaud and Rubio themselves use the notion of computability closure for strengthening their ordering.

In the present paper, we explore the relations between both criteria. We start from the trivial remark that the computability closure itself defines an ordering: $t >_R u$ if $t = f\vec{t}$ and $u \in \mathrm{CC}_R(\vec{t})$. Proving the well-foundedness of this ordering simply consists in proving that the computability closure is correct. Then, we remark that $>_R$ is monotone and continuous for inclusion wrt R. Thus, the computability closure admits a fixpoint which is a well-founded ordering. In the first case order, we prove that this ordering is the recursive path ordering. In the higher-order case, we prove that we get an ordering containing HORPO. Although, we do not get in this case a better definition, it shows that the well-foundedness of HORPO can be reduced to the correctness of the computability closure. This also provide a way to easily strengthen HORPO. Another advantage of this approach is that it can easily be extended to more complex type systems.

2 First-order case

To illustrate our approach, we first begin by presenting the first-order case which is interesting on its own.

We assume given a set $\mathcal{X}$ of variables and a disjoint set $\mathcal{F}$ of function symbols. Let $\mathcal{T}$ be the set of first-order algebraic terms built from $\mathcal{F}$ and $\mathcal{X}$ as usual. Let $\mathcal{V}(t)$ (resp. $\mathcal{F}(t)$) be the set of variables (resp. symbols) occurring in t.

We assume given a *precedence* $\geq_\mathcal{F}$ on $\mathcal{F}$, that is, a quasi-ordering whose strict part $>_\mathcal{F} = \geq_\mathcal{F} \setminus \leq_\mathcal{F}$ is well-founded. Let $\simeq_\mathcal{F} = \geq_\mathcal{F} \cap \leq_\mathcal{F}$ be its associated equivalence relation.

A precedence can be seen as a particular case of quasi-ordering on terms looking at top symbols only. We could extend our results to this more general case, leading to extensions of the semantic path ordering. See [17] for the first-order case, and [10] for the higher-order case.

We assume that every symbol $f \in \mathcal{F}$ is equipped with a *status* $\mathrm{stat}_f \in \{\mathrm{lex}, \mathrm{mul}\}$ defining how the arguments of f must be compared: lexicographically (from left to right, or from right to left) or by multiset. We also assume that $\mathrm{stat}_f = \mathrm{stat}_g$ whenever $f \simeq_{\mathcal{F}} g$.

Definition 1 Given a relation $>$ on terms, let $(f, \vec{t}) >_{\mathrm{stat}} (g, \vec{u})$ iff either $f >_{\mathcal{F}} g$ or $f \simeq_{\mathcal{F}} g$ and $\vec{t} >^{+}_{\mathrm{stat}_f} \vec{u}$.

The ordering $>_{\mathrm{stat}}$ is well-founded whenever $>$ so is ($>_{\mathcal{F}}$ is well-founded).

As usual, the set $\mathrm{Pos}(t)$ of *positions* in a term t is defined as words on positive integers. If $p \in \mathrm{Pos}(t)$, then $t|_p$ is the *subterm* of t at position p, and $t[u]_p$ is the term t with $t|_p$ replaced by u. Let $\trianglelefteq$ be the subterm relation.

A relation $>$ on terms is *stable by substitution* if $t\theta > u\theta$ whenever $t > u$. It is *stable by context* if $C[t]_p > C[u]_p$ whenever $t > u$. It is a *rewrite relation* if it is both stable by substitution and context. Given a relation on terms R, let $\to_R$ be the smallest rewrite relation containing R, R^+ be the transitive closure of R, and $\mathrm{SN}(R)$ be the set of terms that are strongly normalizing for R.

Figure 1: First-order computability closure

$$\text{(arg)} \quad t_i \in \mathrm{CC}^f_R(\vec{t})$$

$$\text{(decomp)} \quad \frac{g\vec{u} \in \mathrm{CC}^f_R(\vec{t})}{u_i \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(prec)} \quad \frac{f >_{\mathcal{F}} g \quad \vec{u} \in \mathrm{CC}^f_R(\vec{t})}{g\vec{u} \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(call)} \quad \frac{f \simeq_{\mathcal{F}} g \quad \vec{u} \in \mathrm{CC}^f_R(\vec{t}) \quad \vec{t}\ (\to^+_R \cup \rhd)_{\mathrm{stat}_f}\ \vec{u}}{g\vec{u} \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(red)} \quad \frac{u \in \mathrm{CC}^f_R(\vec{t}) \quad u \to^+_R v}{v \in \mathrm{CC}^f_R(\vec{t})}$$

Hereafter is a definition of computability closure similar to the one given in [8] except that:

- it is restricted to untyped first-order terms,
- we abstracted away the set R of rules and explicitly put it as argument of the computability closure,
- we added $\to^+_R$ for comparing arguments in (call).

The main novelty is the addition of $\rightarrow_R^+$ in (call). This allows us to get the recursive behavior of RPO: one can use the ordering itself for comparing the arguments of a recursive call. The fact that this is a computability-preserving operation was implicit in [8]. A complete proof of this fact for the higher-order case is given in Lemma 17.

Definition 2 (Computability closure) Let R be a relation on terms. The *computability closure* of a term $f\vec{t}$, written $\mathrm{CC}_R^f(\vec{t})$, is inductively defined in Figure 1. Let $\mathrm{CR}(R)$ be the set of pairs $(f\vec{t}, u)$ such that $u \in \mathrm{CC}_R^f(\vec{t})$.

One can easily prove that CR is monotone and ω-sup-continuous for inclusion. It has therefore a least fixpoint that is reachable by iteration from $\emptyset$.

Definition 3 (Computability ordering) Let the *first-order recursive computability ordering* $>_{\mathrm{rco}}$ be the least fixpoint of CR.

Note that one gets the same ordering by replacing in (red) $\rightarrow_R^+$ by R, and in (call) $\rightarrow_R^+ \cup \rhd$ by R.

Lemma 4 $>_{\mathrm{rco}}$ is a transitive rewrite relation containing subterm.

Proof. Since CR is ω-sup-continuous and preserves the stability by substitution, $>_{\mathrm{rco}}$ is stable by substitution. For the transitivity, assume that $t >_{\mathrm{rco}} u >_{\mathrm{rco}} v$. Then, t must be of the form $f\vec{t}$ and, by (red), $t >_{\mathrm{rco}} v$. For the stability by context, let $v = f\vec{a}t\vec{b}$ and $t >_{\mathrm{rco}} u$. By (arg), $v >_{\mathrm{rco}} \vec{a}t\vec{b}$. By (red), $v >_{\mathrm{rco}} u$. Thus, $\vec{a}t\vec{b}\ (>_{\mathrm{rco}})_{\mathrm{stat}_f}\ \vec{a}u\vec{b}$ and, by (call), $v >_{\mathrm{rco}} f\vec{a}u\vec{b}$. Finally, $>_{\mathrm{rco}}$ contains subterm by (arg). ■

It follows that (decomp) is derivable from (arg) and transitivity. We introduce in Figure 2 an inductive formulation of $>_{\mathrm{rco}}$ obtained by replacing in the rules defining the computability closure $u \in \mathrm{CC}_R^f(\vec{t})$ by $f\vec{t} >_{\mathrm{rco}} u$, and R by $>_{\mathrm{rco}}$.

This simple change in notations clearly shows that $_{\mathrm{rco}}$ is equal to $>_{\mathrm{rpo}}$, whose definition is recalled in Figure 3.

3 Preliminaries to the higher-order case

Before presenting the computability closure for the higher-order case, we first present the ingredients of the termination proof. As explained in the introduction, it is based on an adaptation of Tait's computability technique. First, we interpret each type by a set of computable terms and prove common properties about computable terms. Then, following [6], we define some ordering on computable terms that will be used in the place of the subterm ordering for comparing arguments in recursive calls.

We consider simply-typed λ-terms with curried constants. Let $\mathcal{B}$ be a set of *base types*. The set $\mathbb{T}$ of *simple types* is inductively defined as usual. The set $\mathrm{Pos}(T)$ of *positions in a type* T is defined as usual as words on $\{1, 2\}$. The sets $\mathrm{Pos}^+(T)$ and $\mathrm{Pos}^-(T)$ of *positive and negative positions* respectively are inductively defined as follows:

Figure 2: First-order recursive computability ordering

$$\text{(arg)}\quad f\vec{t} >_{\text{rco}} t_i$$

$$\text{(prec)}\quad \frac{f >_{\mathcal{F}} g \quad f\vec{t} >_{\text{rco}} \vec{u}}{f\vec{t} >_{\text{rco}} g\vec{u}}$$

$$\text{(call)}\quad \frac{f \simeq_{\mathcal{F}} g \quad f\vec{t} >_{\text{rco}} \vec{u} \quad \vec{t}\,(>_{\text{rco}})_{\text{stat}_f}\,\vec{u}}{f\vec{t} >_{\text{rco}} g\vec{u}}$$

$$\text{(red)}\quad \frac{f\vec{t} >_{\text{rco}} u \quad u >_{\text{rco}} v}{f\vec{t} >_{\text{rco}} v}$$

Figure 3: First-order recursive path ordering

$$(1)\quad \frac{t_i \geq_{\text{rpo}} u}{f\vec{t} >_{\text{rpo}} u}$$

$$(2)\quad \frac{f >_{\mathcal{F}} g \quad f\vec{t} >_{\text{rpo}} \vec{u}}{f\vec{t} >_{\text{rpo}} g\vec{u}}$$

$$(3)\quad \frac{f \simeq_{\mathcal{F}} g \quad \vec{t}\,(>_{\text{rpo}})_{\text{stat}_f}\,\vec{u} \quad f\vec{t} >_{\text{rpo}} \vec{u}}{f\vec{t} >_{\text{rpo}} g\vec{u}}$$

– $\text{Pos}^\delta(\mathsf{B}) = \{\varepsilon\}$.
– $\text{Pos}^\delta(T \Rightarrow U) = 1 \cdot \text{Pos}^{-\delta}(T) \cup 2 \cdot \text{Pos}^\delta(U)$.

Let $\text{Pos}(\mathsf{B}, T)$ be the positions of the occurrences of B in T. A base type B *occurs only positively* (resp. *negatively*) in a type T if $\text{Pos}(\mathsf{B}, T) \subseteq \text{Pos}^+(T)$ (resp. $\text{Pos}(\mathsf{B}, T) \subseteq \text{Pos}^-(T)$).

Let $\mathcal{X}$ be a set of *variables* and $\mathcal{F}$ be a disjoint set of *symbols*. We assume that every $a \in \mathcal{X} \cup \mathcal{F}$ is equipped with a type $T_a \in \mathbb{T}$. The sets $\mathcal{T}^T$ of *terms of type* T are inductively defined as follows:

– If $a \in \mathcal{X} \cup \mathcal{F}$, then $a \in \mathcal{T}^{T_a}$.
– If $x \in \mathcal{X}$ and $t \in \mathcal{T}^U$, then $\lambda xt \in \mathcal{T}^{T_x \Rightarrow U}$.
– If $v \in \mathcal{T}^{T \Rightarrow U}$ and $t \in \mathcal{T}^T$, then $vt \in \mathcal{T}^U$.

As usual, we assume that, for all type T, the set of variables of type T is infinite, and consider terms up to type-preserving renaming of bound variables. In the following, $t : T$ or t^T means that $t \in \mathcal{T}^T$. Let $\text{FV}(t)$ be the set of variables *free* in t.

Definition 5 (Accessible arguments) For every $f^{\vec{T} \Rightarrow \mathsf{B}} \in \mathcal{F}$, let $\mathrm{Acc}(f) = \{i \leq |\vec{T}| \mid \mathrm{Pos}(\mathsf{B}, T_i) \subseteq \mathrm{Pos}^+(T_i)\}$.

Definition 6 (Rewrite rules) A *rewrite rule* is a pair of terms (t^T, u^U) such that t is of the form $f\vec{t}$, $\mathrm{FV}(u) \subseteq \mathrm{FV}(t)$ and $T = U$.

In the following, we assume given a set R of rewrite rules. Let $\rightarrow\, = \,\rightarrow_\beta \cup \rightarrow_R$, $\mathrm{SN} = \mathrm{SN}(\rightarrow)$ and $\mathrm{SN}^T = \mathrm{SN} \cap \mathcal{T}^T$. Let $\mathcal{C}$ be the set of symbols c such that, for every rule $(f\vec{t}, u) \in R$, $f \neq c$. The symbols of $\mathcal{C}$ are said *constant*, while the symbols of $\mathcal{D} = \mathcal{F} \setminus \mathcal{C}$ are said *defined*.

3.1 Interpretation of types

Definition 7 (Interpretation of types) A term is *neutral* if it is of the form $x\vec{u}$ or of the form $(\lambda xt)\vec{u}$. Let $\mathcal{Q}_R^T$ be the set of all sets of terms P such that:

(1) $P \subseteq \mathrm{SN}^T$.

(2) P is stable by $\rightarrow$.

(3) If $t : T$ is neutral and $\rightarrow(t) \subseteq P$, then $t \in P$.

Let $\mathcal{I}_R$ be the set of functions I from $\mathcal{B}$ to $\bigcup_{\mathsf{B} \in \mathcal{B}} \mathcal{Q}_R^{\mathsf{B}}$ such that, for all $\mathsf{B} \in \mathcal{B}$, $I(\mathsf{B}) \in \mathcal{Q}_R^{\mathsf{B}}$. Given an *interpretation of base types* $I \in \mathcal{I}_R$, we define an interpretation $[\![T]\!]_R^I \in \mathcal{Q}_R^T$ for any type T as follows:

– $[\![\mathsf{B}]\!]_R^I = I(\mathsf{B})$,

– $[\![T \Rightarrow U]\!]_R^I = \{v \in \mathrm{SN}^{T \Rightarrow U} \mid \forall t \in [\![T]\!]_R^I,\ vt \in [\![U]\!]_R^I\}$.

We also let $F_R^I(\mathsf{B}) = \{t \in \mathrm{SN}^B \mid \forall f^{\vec{T} \Rightarrow \mathsf{B}} \vec{t},\ t \rightarrow^* f\vec{t} \Rightarrow \forall i \in \mathrm{Acc}(f),\ t_i \in [\![T_i]\!]_R^I\}$.

Ordered point-wise by inclusion, $\mathcal{I}_R$ is a complete lattice.

Lemma 8 F_R is a monotone function on $\mathcal{I}_R$.

Proof. We first prove that $P = F_R^I(\mathsf{B}) \in \mathcal{Q}_R^{\mathsf{B}}$.

(1) $P \subseteq \mathrm{SN}^{\mathsf{B}}$ by definition.

(2) Let $t \in P$, $t' \in \rightarrow(v)$, $f : \vec{T} \Rightarrow \mathsf{B}$ and $\vec{t}$ such that $t' \rightarrow^* f\vec{t}$. We must prove that $\vec{t} \in [\![\vec{T}]\!]_R$. It follows from the facts that $t \in P$ and $t \rightarrow^* f\vec{t}$.

(3) Let t^{B} neutral such that $\rightarrow(t) \subseteq P$. Let $f^{\vec{T} \Rightarrow \mathsf{B}}$, $\vec{t}$ such that $t \rightarrow^* f\vec{t}$ and $i \in \mathrm{Acc}(f)$. We must prove that $t_i \in [\![T_i]\!]_R$. Since t is neutral, $t \neq f\vec{t}$. Thus, there is $t' \in \rightarrow(t)$ such that $t' \rightarrow^* f\vec{t}$. Since $t' \in P$, $t_i \in [\![T_i]\!]_R$.

For the monotony, let $\leq^+ \,=\, \leq$ and $\leq^- \,=\, \geq$. Let $I \leq J$ iff, for all B, $I(\mathsf{B}) \subseteq J(\mathsf{B})$. We first prove that $[\![T]\!]_R^I \subseteq^\delta [\![T]\!]_R^J$ whenever $I \leq J$ and $\mathrm{Pos}(\mathsf{B}, T) \subseteq \mathrm{Pos}^\delta(T)$, by induction on T.

– Assume that $T = C \in \mathcal{B}$. Then, $\delta = +$, $[\![T]\!]_R^I = I(C)$ and $[\![T]\!]_R^I = J(C)$. Since $I(C) \subseteq J(C)$, $[\![T]\!]_R^I \subseteq [\![T]\!]_R^I$.

– Assume that $T = U \Rightarrow V$. Then, $\mathrm{Pos}(\mathsf{B}, U) \subseteq \mathrm{Pos}^{-\delta}(U)$ and $\mathrm{Pos}(\mathsf{B}, V) \subseteq \mathrm{Pos}^{\delta}(V)$. Thus, by induction hypothesis, $[\![U]\!]_R^I \subseteq^{-\delta} [\![U]\!]_R^J$ and $[\![V]\!]_R^I \subseteq^{\delta} [\![V]\!]_R^J$. Assume that $\delta = +$. Let $t \in [\![T]\!]_R^I$ and $u \in [\![U]\!]_R^J$. We must prove that $tu \in [\![V]\!]_R^J$. Since $[\![U]\!]_R^I \supseteq [\![U]\!]_R^J$, $tu \in [\![V]\!]_R^I$. Since $[\![V]\!]_R^I \subseteq [\![V]\!]_R^J$, $tu \in [\![V]\!]_R^J$. It works similarly for $\delta = -$.

Assume now that $I \leq J$. We must prove that, for all B, $F_R^I(\mathsf{B}) \subseteq F_R^J(\mathsf{B})$. Let $\mathsf{B} \in \mathcal{B}$ and $t \in F_R^I(\mathsf{B})$. We must prove that $t \in F_R^J(\mathsf{B})$. First, we have $t \in \mathrm{SN}^{\mathsf{B}}$ since $t \in F_R^I(\mathsf{B})$. Assume now that $t \rightarrow^* f^{\vec{T} \Rightarrow \mathsf{B}}\vec{t}$ and let $i \in \mathrm{Acc}(f)$. We must prove that $t_i \in [\![T_i]\!]_R^J$. Since $t \in F_R^I(\mathsf{B})$, $t_i \in [\![T_i]\!]_R^I$. Since $i \in \mathrm{Acc}(f)$, $\mathrm{Pos}(\mathsf{B}, T_i) \subseteq \mathrm{Pos}^+(T_i)$ and $[\![T_i]\!]_R^I \subseteq [\![T_i]\!]_R^J$. ■

Definition 9 (Computability) Let I_R be the least fixpoint of F_R. A term $t : T$ is *R-computable* if $t \in [\![T]\!]_R = [\![T]\!]_R^{I_R}$.

3.2 Computability properties

Lemma 10 If t, u and $t\{x \mapsto u\}$ are computable, then $(\lambda xt)u$ is computable.

Proof. Since $(\lambda xt)u$ is neutral, it suffices to prove that every reduct is computable. Since t and u are SN, we can proceed by induction on (t, u) with $\rightarrow_{\mathrm{lex}}$ as well-founded ordering. Assume that $(\lambda xt)u \rightarrow v$. If $v = t\{x \mapsto u\}$, then t' is computable by assumption. Otherwise, $v = (\lambda xt')u$ with $t \rightarrow t'$, or $v = (\lambda xt)u'$ with $u \rightarrow u'$. In both cases, we can conclude by induction hypothesis. ■

Lemma 11 A term $f\vec{t} : \mathsf{B}$ is computable whenever every reduct of $f\vec{t}$ is computable and, for all $i \in \mathrm{Acc}(f)$, t_i is computable.

Proof. Assume that $f\vec{t} \rightarrow^* g\vec{u}$ with $g : \vec{U} \Rightarrow \mathsf{B}$. Let $i \in \mathrm{Acc}(g)$. If $f\vec{t} \neq g\vec{u}$, then there is $v \in \rightarrow(f\vec{t})$ such that $v \rightarrow^* g\vec{u}$. Since v is computable, u_i is computable. Otherwise, $u_i = t_i$ is computable by assumption. ■

Lemma 12 Every constant symbol is computable.

Proof. Let $c^{\vec{T} \Rightarrow \mathsf{B}} \in \mathcal{C}$ and $\vec{t} \in [\![\vec{T}]\!]_R$. By Lemma 11, $c\vec{t}$ is computable if every reduct of $c\vec{t}$ is computable. Since $\vec{t} \in \mathrm{SN}$, we can proceed by induction on $\vec{t}$ with $\rightarrow_{\mathrm{lex}}$ as well-founded ordering. Assume that $c\vec{t} \rightarrow u$. Since $c \in \mathcal{C}$, $u = c\vec{t'}$ with $\vec{t} \rightarrow_{\mathrm{lex}} \vec{t'}$. Thus, by induction hypothesis, $c\vec{t'}$ is computable. ■

Lemma 13 If every defined symbol is computable, then every term is computable.

Proof. First note that the identity substitution is computable since variables are computable (they are neutral and irreducible). We then prove that, for every term t and computable substitution θ, $t\theta$ is computable, by induction on t.

– Assume that $t = f \in \mathcal{D}$. Then, by assumption, $t\theta = f$ is computable.

– Assume that $t = c \in \mathcal{C}$. Then, by Lemma 12, $t\theta = c$ is computable.

– Assume that $t = x \in \mathcal{X}$. Then, $t\theta = x\theta$ is computable since θ is computable.
– Assume that $t = \lambda xu$. Then, $t\theta = \lambda xu\theta$. Let $v \in [\![V]\!]_R$. We must prove that $t\theta v \in [\![U]\!]_R$. By induction hypothesis, $u\theta\{x \mapsto v\}$ is computable. Since $u\theta$ and v are computable too, by Lemma 10, $t\theta$ is computable.
– Assume that $t = u^{V \Rightarrow T} v$. Then, $t\theta = u\theta v\theta$. By induction hypothesis, $u\theta \in [\![V \Rightarrow T]\!]_R$ and $v\theta \in [\![V]\!]_R$. Thus, $t\theta \in [\![T]\!]_R$. ∎

3.3 Size ordering

The least fixpoint of F_R, I_R, is reachable by transfinite iteration from the smallest element of $\mathcal{I}_R$. This provides us with the following ordering.

Definition 14 (Size ordering) For all $\mathsf{B} \in \mathcal{B}$ and $t \in [\![\mathsf{B}]\!]_R$, let the *size* of t be the smallest ordinal $o_R^{\mathsf{B}}(t) = \mathfrak{a}$ such that $t \in F_R^{\mathfrak{a}}(\emptyset)(\mathsf{B})$, where $F_R^{\mathfrak{a}}$ is the transfinite $\mathfrak{a}$-iteration of F_R. Let $\succ_R = \bigcup_{T \in \mathcal{B}^\Rightarrow} \succ_R^T$, where $(\succ_R^T)_{T \in \mathcal{B}^\Rightarrow}$ is the family of orderings inductively defined as follows:
– For all $\mathsf{B} \in \mathcal{B}$, let $t \succ_R^{\mathsf{B}} u$ iff $t, u \in [\![\mathsf{B}]\!]_R$ and $o_R^{\mathsf{B}}(t) > o_R^{\mathsf{B}}(u)$.
– For all $T, U \in \mathcal{B}^\Rightarrow$, let $t \succ_R^{T \Rightarrow U} u$ iff $t, u \in [\![T \Rightarrow U]\!]_R$ and, for all $v \in [\![T]\!]_R$, $tv \succ_R^U uv$.

In the first-order case, recursive call arguments where compared with the subterm ordering. But the subterm ordering is not adapted to higher-order rewriting. Consider for instance the following simplification rule on process algebra [26]:

$$(\Sigma P); x \rightarrow \Sigma(\lambda y P y; x)$$

where $\Sigma^{(\mathsf{D} \Rightarrow \mathsf{P}) \Rightarrow \mathsf{P}}$ is a data-dependent choice operator and $;^{\mathsf{P} \Rightarrow \mathsf{P} \Rightarrow \mathsf{P}}$ the sequence operator. The term Py is not a subterm of ΣP. The interpretation of P gives us the solution: $[\![\mathsf{P}]\!]_R = \{t \in \mathrm{SN}^P \mid \forall f^{\vec{T} \Rightarrow \mathsf{P}} \vec{t},\ t \rightarrow^* f\vec{t} \Rightarrow \forall i \in \mathrm{Acc}(f),\ t_i \in [\![T_i]\!]_R\}$. Since P occurs only positively in $\mathsf{D} \Rightarrow \mathsf{P}$, $\mathrm{Acc}(\Sigma) = \{1\}$. Hence, if $\Sigma P \in [\![\mathsf{P}]\!]_R$ then, for all $d \in [\![\mathsf{D}]\!]_R$, $pd \in [\![\mathsf{P}]\!]_R$ and $o_R^{\mathsf{P}}(Pd) < o_R^{\mathsf{P}}(\Sigma P)$.

We immediately check that the size ordering is well-founded.

Lemma 15 $\succ_R^T$ is transitive and well-founded.

Proof. By induction on T. For $T \in \mathcal{B}$, this is immediate. Assume now that $(t_i)_{i \in \mathbb{N}}$ is an increasing sequence for $\succ_R^{T \Rightarrow U}$. Since variables are computable, let $x \in [\![T]\!]_R$. By definition of $\succ_R^{T \Rightarrow U}$, $(t_i x)_{i \in \mathbb{N}}$ is an increasing sequence for $\succ_R^U$. ∎

In case of a first-order type B, when $\rightarrow$ is confluent, the size of t^{B} is the number of (constructor) symbols at the top of its normal form. So, it is equivalent to using embedding on normal forms. But, since the ordering is compatible with reduction, in the sense that $t \succeq_R u$ whenever $t \rightarrow u$, it is finer than the embedding. For instance, by taking the rules:

$$\begin{array}{rcl} x - 0 & \rightarrow & x \\ 0 - x & \rightarrow & 0 \\ (sx) - (sy) & \rightarrow & x - y \end{array}$$

one can prove that $t - u \trianglelefteq_R t$. This allows to prove the termination of functions for which simplification orderings fail like:

$$\begin{array}{rcl} 0/y & \rightarrow & 0 \\ (sx)/y & \rightarrow & s((x-y)/y) \end{array}$$

However, in practice, the size ordering cannot be used as is. We need a decidable syntactic approximation. In [6], we assume given an ordered term algebra $(\mathcal{A}, >_\mathcal{A})$ for representing operations on ordinals and, for each base type B and expression $a \in \mathcal{A}$, we introduce the subtype B^a of terms of type B whose size is less than or equal to a. Then, in the (call) rule, the size annotations of $\vec{t}$ and $\vec{u}$ are compared with $>_\mathcal{A}$. In [1], we prove that type checking is decidable, whenever the constraints generated by these comparisons are satisfiable, hence providing a powerful termination criterion. We do not use size annotations here, but it would definitely be a natural and powerful extension. Instead, we are going to define an approximation like in [7].

4 Higher-order case

We now introduce the size-ordering approximation and the computability closure for the higher-order case.

Definition 16 (Computability closure) The *computability closure* of a term $f\vec{t}$, written $\mathrm{CC}_R^f(\vec{t})$, and the associated size-ordering approximation, written $\rhd_R^{f\vec{t}}$, are mutually inductively defined in Figures 5 and 4 respectively. Let $\mathrm{CR}(R)$ be the set of pairs $(f\vec{t}, u)$ such that $u \in \mathrm{CC}_R^f(\vec{t})$, $\mathrm{FV}(u) \subseteq \mathrm{FV}(f\vec{t})$ and $f\vec{t}$ and u have the same type.

Compared to the first-order case, we added the rules (var) and (lam) to build abstractions and, in (call), we replaced $\rightarrow_R^+$ by $\rightarrow_{\beta R}^+$, and $\rhd$ by $\rhd_R^{f\vec{t}}$. This ordering is a better approximation of the size ordering than the one given in [7] where, in ($\rhd$base), $\vec{b} \in \mathcal{X} \setminus \mathrm{FV}(\vec{t})$. In this case, the size-ordering approximation can be defined independently of the computability closure. Note however that, in both cases, the size-ordering approximation contains the subterms of same type. In the process algebra example, by ($\rhd$base), we have $\Sigma P \rhd_R^l Py$ where $l = (\Sigma P); x$.

We now prove the correctness of the computability closure.

Lemma 17 If $R \subseteq \mathrm{CR}(R)$, then $\rightarrow_\beta \cup \rightarrow_{\mathrm{CR}(R)}$ is well-founded.

Proof. Let $S = \mathrm{CR}(R)$. It suffices to prove that every term is S-computable. Let $\rightarrow\, = \,\rightarrow_\beta \cup \rightarrow_S$ and $\mathrm{SN} = \mathrm{SN}(\rightarrow)$. After Lemma 13, it suffices to prove that, for all $f^{\vec{V} \Rightarrow B}$ and $\vec{v} \in [\![\vec{V}]\!]_S$, $f\vec{v} \in [\![B]\!]_S$. We prove it by induction on $((f, \vec{v}), \vec{v})$ with $((\succ_S)_{\mathrm{stat}}, \rightarrow_{\mathrm{lex}})$ as well-founded ordering ($\vec{v}$ are computable) (H1). By Lemma 11, it suffices to prove that $\rightarrow(f\vec{v}) \subseteq [\![B]\!]_S$. Let $v' \in \rightarrow(f\vec{v})$. Either $v' = f\vec{v}'$ with $\vec{v} \rightarrow_{\mathrm{stat}_f} \vec{v}'$, or $v = f\vec{t}\sigma$, $v' = u\sigma$

Figure 4: Higher-order computability closure

$$\text{(arg)}\quad t_i \in \mathrm{CC}^f_R(\vec{t})$$

$$\text{(decomp)}\quad \frac{g\vec{u} \in \mathrm{CC}^f_R(\vec{t}) \quad i \in \mathrm{Acc}(g)}{u_i \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(prec)}\quad \frac{f >_{\mathcal{F}} g}{g \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(call)}\quad \frac{f \simeq_{\mathcal{F}} g^{\vec{U}\Rightarrow U} \quad \vec{u}^{\vec{U}} \in \mathrm{CC}^f_R(\vec{t}) \quad \vec{t}\ (\rightarrow^+_{\beta R} \cup \rhd^{f\vec{t}}_R)_{\mathrm{stat}_f}\ \vec{u}}{g\vec{u} \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(red)}\quad \frac{u \in \mathrm{CC}^f_R(\vec{t}) \quad u \rightarrow^+_{\beta R} v}{v \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(app)}\quad \frac{u^{V\Rightarrow T} \in \mathrm{CC}^f_R(\vec{t}) \quad v^V \in \mathrm{CC}^f_R(\vec{t})}{uv \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(var)}\quad \frac{x \notin \mathrm{FV}(\vec{t})}{x \in \mathrm{CC}^f_R(\vec{t})}$$

$$\text{(lam)}\quad \frac{u \in \mathrm{CC}^f_R(\vec{t}) \quad x \notin \mathrm{FV}(\vec{t})}{\lambda xu \in \mathrm{CC}^f_R(\vec{t})}$$

and $u \in \mathrm{CC}^f_R(\vec{t})$. In the former case, $\vec{v}' \in [\![\vec{V}]\!]_S$ since $[\![\vec{V}]\!]_S$ is stable by $\rightarrow$, and $\vec{v}(\unrhd_S)_{\mathrm{stat}_f}\vec{v}'$. Thus, we can conclude by (H1). For the latter case, we prove that, if $u \in \mathrm{CC}^f_R(\vec{t})$ then, for all S-computable substitution θ such that $\mathrm{dom}(\theta) \subseteq \mathrm{FV}(u) \setminus \mathrm{FV}(\vec{t})$, $u\sigma\theta$ is S-computable, by induction on $\mathrm{CC}^f_R(\vec{t})$ (H2).

(arg) $t_i\sigma = v_i$ is computable by assumption.

(decomp) By (H2), $g\vec{u}\sigma\theta$ is computable. Thus, by definition of I_S, $u_i\sigma\theta$ is computable.

(prec) By (H1), g is computable.

(call) By (H2), $\vec{u}\sigma\theta$ are computable. Since $\mathrm{dom}(\theta) \cap \mathrm{FV}(\vec{t}) = \emptyset$, $t_i\sigma\theta = t_i\sigma = v_i$. Assume that $t_i \rightarrow^+_{\beta R} u_j$. Then, $v_i \rightarrow^+_{\beta R} u_j\sigma\theta$. Since $R \subseteq S$ and $\rightarrow^+_{\beta S} \subseteq \unrhd_S$, $v_i \unrhd_S u_j\sigma\theta$. Assume now that $t_i \rhd^f_R u_j$. We prove that, if $a \rhd^{f\vec{t}}_R b$ then, for all S-computable substitution θ such that $\mathrm{dom}(\theta) \subseteq \mathrm{FV}(b) \setminus (\mathrm{FV}(a) \cup \mathrm{FV}(\vec{t}))$ and $a\sigma\theta$ is S-computable, $b\sigma\theta$ is S-computable and $a\sigma\theta \succ_S b\sigma\theta$.

Figure 5: Ordering for comparing function arguments

$$(\rhd\text{base})\quad \frac{i \in \mathrm{Acc}(g) \quad \vec{b} \in \mathrm{CC}^f_R(\vec{t})}{g^{\vec{A}\Rightarrow B}\vec{a}^{\vec{A}} \rhd^{f\vec{t}}_R a_i^{\vec{B}\Rightarrow B}\vec{b}^{\vec{B}}}$$

$$(\rhd\text{lam})\quad \frac{a \rhd^{f\vec{t}}_R bx \quad x \notin \mathrm{FV}(b) \cup \mathrm{FV}(\vec{t})}{\lambda xa \rhd^{f\vec{t}}_R b}$$

$$(\rhd\text{red})\quad \frac{a \rhd^{f\vec{t}}_R b \quad b \rightarrow^+_{\beta R} c}{a \rhd^{f\vec{t}}_R c}$$

$$(\rhd\text{trans})\quad \frac{a \rhd^{f\vec{t}}_R b \quad b \rhd^{f\vec{t}}_R c}{a \rhd^{f\vec{t}}_R c}$$

($\rhd$base) Let $a = g^{\vec{A}\Rightarrow B}\vec{a}^{\vec{A}}$ and $b = a_i^{\vec{B}\Rightarrow B}\vec{b}^{\vec{B}}$. Let $I^{\mathfrak{a}}_S = F^{\mathfrak{a}}_S(\emptyset)$. Note that the size of a term is necessarily a successor ordinal. Thus, $o_S(a\sigma\theta) = \mathfrak{a} + 1$ and, by definition of $[\![B]\!]_S$, $a_i\sigma\theta \in [\![\vec{B} \Rightarrow B]\!]^{I^{\mathfrak{a}}_S}_S$. Since $\vec{b} \in \mathrm{CC}^f_R(\vec{t})$ and $\mathrm{dom}(\theta) \subseteq \mathrm{FV}(\vec{b}) \setminus \mathrm{FV}(\vec{t})$, by (H2), $\vec{b}\sigma\theta$ are computable. Therefore, $a_i\sigma\theta\vec{b}\sigma\theta \in I^{\mathfrak{a}}_S(B)$ and $o_S(b\sigma\theta) \le \mathfrak{a} < o_S(a\sigma\theta)$.

($\rhd$lam) Let $w \in [\![T_x]\!]_S$. We must prove that $b\sigma\theta w$ is computable. Since $x \notin \mathrm{FV}(b) \cup \mathrm{FV}(\vec{t})$, $x \notin \mathrm{dom}(\sigma\theta)$. W.l.o.g., we can assume that $x \notin \mathrm{codom}(\sigma\theta)$. Thus, $(\lambda xa)\sigma\theta = \lambda xa\sigma\theta$. Let $\theta' = \theta \cup \{x \mapsto w\}$. Since $\lambda xa\sigma\theta$ is computable, $a\sigma\theta'$ is computable. Since $\mathrm{dom}(\theta') \subseteq \mathrm{FV}(bx) \setminus (\mathrm{FV}(a) \cup \mathrm{FV}(\vec{t}))$, by induction hypothesis, $(bx)\sigma\theta' = b\sigma\theta' w$ is computable and $a\sigma\theta' \succ_S b\sigma\theta' w$. Since $x \notin \mathrm{dom}(\sigma)$, $b\sigma\theta' = b\sigma\theta$. Thus, $b\sigma\theta$ is computable and $(\lambda xa)\sigma\theta \succ_S b\sigma\theta$.

($\rhd$red) By induction hypothesis and since $\rightarrow^+_{\beta S} \subseteq \unrhd_S$.

($\rhd$trans) By induction hypothesis and transitivity of $\succ_S$.

Hence, $v_i = t_i\sigma\theta \succ_S u_j\sigma\theta$ since $\mathrm{dom}(\theta) \subseteq \mathrm{FV}(u_j) \setminus (\mathrm{FV}(t_i) \cup \mathrm{FV}(\vec{t}))$ and v_i is computable. Therefore, either $\vec{v}(\succ_S)_{\mathrm{stat}_f}\vec{u}\sigma\theta$ or $\vec{v} \rightarrow^+_{\mathrm{stat}_f} \vec{u}\sigma\theta$ and, by (H1), $f\vec{u}\sigma\theta$ is computable.

(red) By (H2), $u\sigma\theta \in [\![U]\!]_S$. Since $\rightarrow^+_{\beta R}$ is stable by substitution, $u\sigma\theta \rightarrow^+_{\beta R} v\sigma\theta$. Since $R \subseteq S$, $u\sigma\theta \rightarrow^+ v\sigma\theta$. Since $[\![U]\!]_S$ is stable by $\rightarrow$, $v\sigma\theta$ is computable.

(app) By (H1), $u\sigma\theta$ and $v\sigma\theta$ are computable. Thus, by definition of $[\![V \Rightarrow T]\!]_S$, $u\sigma\theta v\sigma\theta$ is computable.

(lam) W.l.o.g, we can assume that $x \notin \mathrm{dom}(\theta) \cup \mathrm{codom}(\sigma\theta)$. Thus, $(\lambda xu)\sigma\theta = \lambda xu\sigma\theta$. Let $v : T_x$ computable and $\theta' = \theta \cup \{x \mapsto v\}$. If $x \notin \mathrm{FV}(u)$, then $u\sigma\theta' = u\sigma\theta$ is computable. Otherwise, since $\mathrm{dom}(\theta') = \mathrm{dom}(\theta) \cup \{x\}$, $\mathrm{dom}(\theta) \subseteq \mathrm{FV}(\lambda xu) \setminus \mathrm{FV}(\vec{t})$ and $x \notin \mathrm{FV}(\vec{t})$, we have $\mathrm{dom}(\theta') \subseteq \mathrm{FV}(u) \setminus \mathrm{FV}(\vec{t})$. Thus, by (H2), $u\theta'$ is computable. Hence, by Lemma 10, $\lambda xu\theta$ is computable.

(var) Since $x \notin \mathrm{FV}(\vec{t})$, $x\sigma\theta = x\theta$ is computable by assumption on θ. ■

Like in the first-order case, one can easily check that the functions $\rhd^{f\vec{t}}$, $\mathrm{CC}^f(\vec{t})$ and CR are monotone and ω-sup-continuous for inclusion.

Definition 18 (Higher-order recursive computability ordering) Let the *weak higher-order recursive computability ordering* $>_{\mathrm{whorco}}$ be the least fixpoint of CR, and the *higher-order recursive computability ordering* $>_{\mathrm{horco}}$ be the closure by context of $>_{\mathrm{whorco}}$.

In the following, let $\rhd_{\mathrm{whorco}} = \rhd_{>_{\mathrm{whorco}}}$ and $\mathrm{CC} = \mathrm{CC}_{>_{\mathrm{whorco}}}$. The well-foundedness of $\rightarrow_\beta \cup >_{\mathrm{horco}}$ immediately follows from Lemma 17 and the facts that $>_{\mathrm{whorco}} \subseteq \mathrm{CR}(>_{\mathrm{whorco}})$ and $\rightarrow_{>_{\mathrm{whorco}}} = >_{\mathrm{horco}}$.

Theorem 19 $\rightarrow_\beta \cup >_{\mathrm{horco}}$ is a well-founded rewrite relation.

Before comparing $>_{\mathrm{horco}}$ with the monomorphic version of $>_{\mathrm{horpo}}$ [14] whose definition is recalled in Figure 6, let us give some examples.

Example 1 (Differentiation) Taken from [9] (Example 10 in [16]). Consider the symbols 0^{R}, 1^{R}, $+^{\mathsf{R}\Rightarrow\mathsf{R}\Rightarrow\mathsf{R}}$, $\times^{\mathsf{R}\Rightarrow\mathsf{R}\Rightarrow\mathsf{R}}$, and $D^{(\mathsf{R}\Rightarrow\mathsf{R})\Rightarrow\mathsf{R}\Rightarrow\mathsf{R}}$. The rule:

$$D\lambda xFx \times Gx \rightarrow \lambda xDFx \times Gx + Fx \times DGx$$

is both in $>_{\mathrm{horco}}$ and $>_{\mathrm{horpo}}$. Take $D >_{\mathcal{F}} \times, +$. By (prec), $t = D\lambda xFx \times Gx > +, \times$. By (var), $t > x$. By (arg), $t > \lambda xFx \times Gx$. By (app), $t > (\lambda xFx \times Gx)x$. By (red), $t > Fx \times Gx$. Since $\mathrm{Acc}(\times) = \{1,2\}$, by (decomp), $t > Fx, Gx$. By ($\rhd$base), $Fx \times Gx \rhd Fx, Gx$. By ($\rhd$lam), $\lambda xFx \times Gx \rhd F, G$. By (call), $t > DF, DG$. By several applications of (app), $t > DFx \times Gx + Fx \times DGx$. Finally, by (abs), $t > \lambda xDFx \times Gx + Fx \times DGx$.

We now give two examples included in $>_{\mathrm{horco}}$ but not in $>_{\mathrm{horpo}}$.

Example 2 (Process Algebra) Taken from [26] (Example 5 in [14]). The rule:

$$(\Sigma P); x \rightarrow \Sigma(\lambda y Py; x)$$

is in $>_{\mathrm{horco}}$ but not in $>_{\mathrm{horpo}}$. Take $\Sigma <_{\mathcal{F}}\, ;$ and $\mathrm{stat}_; = \mathrm{lex}$. By (arg), $t = (\Sigma P); x > \Sigma P, x$. Since $\mathrm{Acc}(\Sigma) = \{1\}$, by (decomp), $t > P$. By (var), $t > y$. By (app), $t > Py$. By ($\rhd$base), $\Sigma P \rhd Py$. By (call), $t > Py; x$. By (lam), $t > \lambda y Py; x$. Thus, by (prec), $t > \Sigma\lambda y Py; x$.

Example 3 (Lists of functions) This is Example 6 in [14]. Consider the symbols $\mathit{fcons}^{(\mathsf{B}\Rightarrow\mathsf{B})\Rightarrow\mathsf{L}\Rightarrow\mathsf{L}}$ and $\mathit{lapply}^{\mathsf{B}\Rightarrow\mathsf{L}\Rightarrow\mathsf{B}}$. The rule:

$$\mathit{lapply}\, x\, (\mathit{fcons}\, F\, l) \rightarrow F\, (\mathit{lapply}\, x\, l)$$

is in $>_{\mathrm{horco}}$ but not in $>_{\mathrm{horpo}}$. Take $\mathrm{stat}_{\mathit{lapply}} = \mathrm{lex}$ (from right to left). By (arg), $t = \mathit{lapply}\, x\, (\mathit{fcons}\, F\, l) > x, \mathit{fcons}\, F\, l$. Since $\mathrm{Acc}(\mathit{fcons}) = \{1,2\}$, by (decomp), $t > F, l$. By ($\rhd$base), $\mathit{fcons}\, F\, l \rhd l$. By (call), $t > \mathit{lapply}\, x\, l$. Thus, by (app), $t > F\, (\mathit{lapply}\, x\, l)$.

5 Comparison with HORPO

Before proving that $>_{\text{horpo}} \subseteq >^+_{\text{horco}}$, we study some properties of $>_{\text{horco}}$.

Lemma 20 (1) $>_{\text{whorco}}$ is stable by substitution.
(2) $>_{\text{whorco}}\rightarrow^+ \subseteq >_{\text{whorco}}$.
(3) If $t >_{\text{whorco}} u$, then $t\vec{w} >_{\text{whorco}} u\vec{w}$.
(4) If $t \rightarrow^+ u$, then $f\vec{a}t\vec{b} >_{\text{whorco}} f\vec{a}u\vec{b}$.
(5) $>_{\text{whorco}}$ is transitive.
(6) $>^+_{\text{horco}}>_{\text{whorco}} \subseteq >_{\text{whorco}}$.

From (2) and (6), it follows that any sequence of $>_{\text{horco}}$-steps with at least one $>_{\text{whorco}}$-step, in fact corresponds to a $>_{\text{whorco}}$-step. So, $>_{\text{horco}}$ is not far from being transitive.

Figure 6: HORPO [14]

$$P(f,\vec{t},u) = f\vec{t} >_{\text{horpo}} u \vee (\exists j)\ t_j \geq_{\text{horpo}} u$$

$$(1)\quad \frac{t_i \geq_{\text{horpo}} u}{f^{\vec{T}\Rightarrow T}\vec{t}^{\vec{T}} >_{\text{horpo}} u^T}$$

$$(2)\quad \frac{f >_{\mathcal{F}} g \quad P(f,\vec{t},\vec{u})}{f^{\vec{T}\Rightarrow T}\vec{t}^{\vec{T}} >_{\text{horpo}} g^{\vec{U}\Rightarrow T}\vec{u}^{\vec{U}}}$$

$$(3)\quad \frac{f \simeq_{\mathcal{F}} g \quad \text{stat}_f = \text{mul} \quad \vec{t}\ (>_{\text{horpo}})_{\text{stat}_f}\ \vec{u}}{f^{\vec{T}\Rightarrow T}\vec{t}^{\vec{T}} >_{\text{horpo}} g^{\vec{U}\Rightarrow T}\vec{u}^{\vec{U}}}$$

$$(4)\quad \frac{f \simeq_{\mathcal{F}} g \quad \text{stat}_f = \text{lex} \quad \vec{t}\ (>_{\text{horpo}})_{\text{stat}_f}\ \vec{u} \quad P(f,\vec{t},\vec{u})}{f^{\vec{T}\Rightarrow T}\vec{t}^{\vec{T}} >_{\text{horpo}} g^{\vec{U}\Rightarrow T}\vec{u}^{\vec{U}}}$$

$$(5)\quad \frac{P(f,\vec{t},\vec{u})}{f^{\vec{T}\Rightarrow T}\vec{t} >_{\text{horpo}} \vec{u}^T}$$

$$(6)\quad \frac{\{t_1,t_2\}\ (>_{\text{horpo}})_{\text{mul}}\ \{u_1,u_2\}}{t_1^{U\Rightarrow T}t_2^U > u_1^{V\Rightarrow T}u_2^V}$$

$$(7)\quad \frac{t >_{\text{horpo}} u}{\lambda xt >_{\text{horpo}} \lambda xu}$$

We now compare $>_{\text{horco}}$ with the monomorphic version of $>_{\text{horpo}}$ defined in Figure 6. For the case (6), let us list all the cases that may be possible *a priori*:

(a) $t_1 \geq_{\mathrm{horpo}} u_1$ and $t_1 \geq_{\mathrm{horpo}} u_2$. This case is not possible since then we would have $U \Rightarrow T = V \Rightarrow T = V$.

(b) $t_2 \geq_{\mathrm{horpo}} u_1$ and $t_2 \geq_{\mathrm{horpo}} u_2$. This case is not possible since then we would have $U = V \Rightarrow T = V$.

(c) $t_1 \geq_{\mathrm{horpo}} u_1$ and $t_2 \geq_{\mathrm{horpo}} u_2$. This case is possible.

(d) $t_2 \geq_{\mathrm{horpo}} u_1$ and $t_1 \geq_{\mathrm{horpo}} u_2$. This case is not possible since then we would have $U = V \Rightarrow T$ and $U \Rightarrow T = V$, and thus $U = (U \Rightarrow T) \Rightarrow T$.

Hence, only (c) is in fact possible. We now prove that $>_{\mathrm{horpo}} \subseteq >^+_{\mathrm{horco}}$.

Theorem 21 $>_{\mathrm{horpo}} \subseteq >^+_{\mathrm{horco}}$.

Proof. We first prove that $f\vec{t} > v$ whenever $f\vec{t} >^+_{\mathrm{horco}} v$ or $t_j >^*_{\mathrm{horco}} v$ (*). Assume that $t_j >^*_{\mathrm{horco}} v$. By (arg), $f\vec{t} > t_j$. Thus, by (red), $f\vec{t} > v$. Assume now that $f\vec{t} >_{\mathrm{horco}} u >^*_{\mathrm{horco}} v$. There are 2 cases:

– $f\vec{t} = f\vec{a}t_k\vec{b}$, $u = f\vec{a}t'_k\vec{b}$ and $t_k >_{\mathrm{horco}} t'_k$. By Lemma 20 (4), $f\vec{t} >_{\mathrm{whorco}} u$. By Lemma 20 (2), $f\vec{t} >_{\mathrm{whorco}} v$. Thus, $f\vec{t} > v$.

– $f\vec{t} = f\vec{l}\sigma\vec{b}$, $u = r\sigma\vec{b}$ and $f\vec{l}\sigma >_{\mathrm{whorco}} r\sigma$. By Lemma 20 (3), $f\vec{t} >_{\mathrm{whorco}} u$. By Lemma 20 (2), $f\vec{t} >_{\mathrm{whorco}} v$. Thus, $f\vec{t} > v$.

We now prove the theorem by induction on $>_{\mathrm{horpo}}$.

(1) By induction hypothesis, $t_i >^*_{\mathrm{horco}} u$. By (arg), $f\vec{t} > t_i$. Since $t_i >_{\mathrm{horpo}} u$ and $f\vec{t} >_{\mathrm{horpo}} u$, $(f\vec{t}, t_i)$ is a rule. Thus, $f\vec{t} >_{\mathrm{whorco}} t_i$ and, by Lemma 20 (2), $f\vec{t} >_{\mathrm{whorco}} u$.

(2) By induction hypothesis, for all i, $f\vec{t} >^+_{\mathrm{horco}} u_i$ or $t_j >^*_{\mathrm{horco}} u_i$. Hence, by (*), $f\vec{t} > \vec{u}$. By (prec), $f\vec{t} > g$. Thus, by (app), $f\vec{t} > g\vec{u}$. Since $(f\vec{t}, g\vec{u})$ is a rule, $f\vec{t} >_{\mathrm{whorco}} g\vec{u}$.

(3) By induction hypothesis, $\vec{t}\ (>^+_{\mathrm{horco}})_{\mathrm{mul}}\ \vec{u}$. Hence, by (*), $f\vec{t} > \vec{u}$. Thus, by (call), $f\vec{t} > g\vec{u}$. Since $(f\vec{t}, g\vec{u})$ is a rule, $f\vec{t} >_{\mathrm{whorco}} g\vec{u}$.

(4) By induction hypothesis, $\vec{t}\ (>^+_{\mathrm{horco}})_{\mathrm{stat}_f}\ \vec{u}$ and, for all i, $f\vec{t} >^+_{\mathrm{horco}} u_i$ or $t_j >^*_{\mathrm{horco}} u_i$. Hence, by (*), $f\vec{t} > \vec{u}$. Thus, by (call), $f\vec{t} > g\vec{u}$. Since $(f\vec{t}, g\vec{u})$ is a rule, $f\vec{t} >_{\mathrm{whorco}} g\vec{u}$.

(5) By induction hypothesis, for all i, $f\vec{t} >^+_{\mathrm{horco}} u_i$ or $t_j >^*_{\mathrm{horco}} u_i$. Hence, by (*), $f\vec{t} > u_i$ for all i. Thus, by (app), $f\vec{t} > \vec{u}$. Since $(f\vec{t}, \vec{u})$ is a rule, $f\vec{t} >_{\mathrm{whorco}} \vec{u}$.

(6) As previously remarked, $t_1 \geq_{\mathrm{horpo}} u_1$ and $t_2 \geq_{\mathrm{horpo}} u_2$. Thus, by induction hypothesis, $t_1 >^*_{\mathrm{horco}} u_1$ and $t_2 >^*_{\mathrm{horco}} u_2$. Hence, by monotony, $t_1t_2 >^*_{\mathrm{horco}} u_1t_2 >^*_{\mathrm{horco}} u_1u_2$.

(7) By induction hypothesis, $t >_{\mathrm{horco}} u$. Thus, by context, $\lambda xt >_{\mathrm{horco}} \lambda xu$. ■

From the proof, we observe that, if (6) were restricted to $(t_1 >_{\mathrm{horpo}} u_1 \wedge t_2 = u_2) \vee (t_1 = u_1 \wedge t_2 >_{\mathrm{horpo}} u_2)$, then we would get $>_{\mathrm{horpo}} \subseteq >_{\mathrm{horco}}$, since this is the only case requiring transitivity.

In [14], the authors strengthen their definition of HORPO by adding in $P(f, \vec{t}, \vec{u})$ the case $u_i \in \mathcal{CC}(f\vec{t})$, where $\mathcal{CC}(f\vec{t})$ is similar to $\mathrm{CC}^f_\emptyset(\vec{t})$ with the subterm ordering $\rhd$ instead of $\rhd^f$ in (call). Thus, (*) is still satisfied and $>_{\mathrm{horpo}} \subseteq >^+_{\mathrm{horco}}$ in this case too.

In [16], the authors add a few new cases to HORPO and extend the computability closure a little bit. But, again, this does not make any essential difference. And, indeed, they recognize they are not satisfied with their treatment of abstractions. Taking our interpretation of base types solve these problems.

6 Conclusion

We proved that the recursive path ordering is strictly included (equal in the first-order case) to the recursive computability ordering, an ordering naturally defined from the notion of computability closure. In the higher-order case, this does not provide us with a very practical definition. However, the well-foundedness proof is reduced to proving the correctness of the computability closure. This therefore provides us with a way to easily extend HORPO to richer type systems. For instance, in [7], we proved the correctness of the computability closure for a polymorphic and dependent type system with both object and type level rewriting. This would generalize Walukiewicz' extension of HORPO [28]. In [3], we defined an extension of the computability closure accepting non-simply terminating systems. Finally, in [4], we proved that the computability closure proves the termination of rewriting modulo AC as well.

References

[1] F. Blanqui. Decidability of type-checking in the Calculus of Algebraic Constructions with size annotations. In *Proc. of CSL'05*, LNCS 3634.

[2] F. Blanqui. Definitions by rewriting in the Calculus of Constructions (extended abstract). In *Proc. of LICS'01*.

[3] F. Blanqui. Inductive types in the Calculus of Algebraic Constructions. In *Proc. of TLCA'03*, LNCS 2701.

[4] F. Blanqui. Rewriting modulo in Deduction modulo. In *Proc. of RTA'03*, LNCS 2706.

[5] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Proc. of RTA'00*, LNCS 1833.

[6] F. Blanqui. A type-based termination criterion for dependently-typed higher-order rewrite systems. In *Proc. of RTA'04*, LNCS 3091.

[7] F. Blanqui. Definitions by rewriting in the Calculus of Constructions. *Mathematical Structures in Computer Science*, 15(1):37–92, 2005.

[8] F. Blanqui, J.-P. Jouannaud, and M. Okada. The Calculus of Algebraic Constructions. In *Proc. of RTA'99*, LNCS 1631.

[9] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive-data-type Systems. *Theoretical Computer Science*, 272:41–68, 2002.

[10] C. Borralleras and A. Rubio. A monotonic higher-order semantic path ordering. In *Proc. of LPAR'01*, LNCS 2250.

[11] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[12] J.-P. Jouannaud and M. Okada. Executable higher-order algebraic specification languages. In *Proc. of LICS'91*.

[13] J.-P. Jouannaud and M. Okada. Abstract Data Type Systems. *Theoretical Computer Science*, 173(2):349–391, 1997.

[14] J.-P. Jouannaud and A. Rubio. The Higher-Order Recursive Path Ordering. In *Proc. of LICS'99*.

[15] J.-P. Jouannaud and A. Rubio. A recursive path ordering for higher-order terms in eta-long beta-normal form. In *Proc. of RTA'96*, LNCS 1103.

[16] J.-P. Jouannaud and A. Rubio. Higher-order recursive path orderings *"à la carte"*, 2001. Draft.

[17] S. Kamin and J.-J. Lévy. Two generalizations of the Recursive Path Ordering, 1980. Unpublished.

[18] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.

[19] J. B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95:210–225, 1960.

[20] C. Loria-Saenz and J. Steinbach. Termination of combined (rewrite and λ-calculus) systems. In *Proc. of CTRS'92*, LNCS 656.

[21] O. Lysne and J. Piris. A termination ordering for higher order rewrite systems. In *Proc. of RTA'95*, LNCS 914.

[22] R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192(2):3–29, 1998.

[23] D. Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In *Proc. of ELP'89*, LNCS 475.

[24] D. A. Plaisted. A recursively defined ordering for proving termination of term rewriting systems. Technical report, University of Illinois, Urbana-Champaign, United States, 1978.

[25] W. W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Proceedings of the 1972 Logic Colloquium*, volume 453 of *Lecture Notes in Mathematics*, 1975.

[26] J. van de Pol. Termination proofs for higher-order rewrite systems. In *Proc. of HOA '93*, LNCS 816.

[27] V. van Oostrom and F. van Raamsdonk. Comparing Combinatory Reduction Systems and Higher-order Rewrite Systems. In *Proc. of HOA '93*, LNCS 816.

[28] D. Walukiewicz-Chrząszcz. Termination of rewriting in the Calculus of Constructions. *Journal of Functional Programming*, 13(2):339–414, 2003.

[29] D. Walukiewicz-Chrząszcz. *Termination of Rewriting in the Calculus of Constructions.* PhD thesis, Warsaw University, Poland and Université d'Orsay, France, 2003.

Contents

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399